NXP Semiconductors
Reference Manual

Document identifier: MCXNP184M150F70RM
Rev. 7, 2025-10-10

MCX N Reference Manual

Supports MCX N94, N54, N53, N52 and N24

Contents

Chapter 1 About This Manual..... 15
1.1 About This Document..... 15

Chapter 2 Introduction..... 18
2.1 Overview..... 18
2.2 Target applications..... 19
2.3 Block diagram..... 21
2.4 System bus priority and arbitration..... 22
2.5 Functional overview..... 23

Chapter 3 Core Overview..... 36
3.1 Introduction..... 36
3.2 CPU0 and CPU1 Cortex-M33 Code and System buses..... 36
3.3 Nested Vectored Interrupt Controller (NVIC)..... 37
3.4 Implementation Defined Attribution Unit (IDAU)..... 44

Chapter 4 Memory..... 45
4.1 Memory architecture..... 45
4.2 System memory map..... 45
4.3 Cache..... 48
4.4 SRAM..... 48
4.5 Read Only Memory (ROM)..... 51
4.6 Internal flash..... 51
4.7 External memory..... 53
4.8 Peripheral Bridge (PBRG)..... 53

Chapter 5 Low-Power Cache Controller (LPCAC)..... 62
5.1 Chip-specific LPCAC information..... 62
5.2 Overview..... 63
5.3 Functional description..... 63
5.4 Signal descriptions..... 64
5.5 Memory regions and input control description..... 65

Chapter 6 CACHE64 Controller (CACHE64_CTRL)..... 67
6.1 Chip-specific CACHE64_CTRL information..... 67
6.2 Overview..... 68
6.3 Functional description..... 68
6.4 External signals..... 74
6.5 Initialization..... 74
6.6 Memory map and registers..... 75

Chapter 7 CACHE64 Policy Select (CACHE64_POLSEL)..... 83
7.1 Chip-specific CACHE64_POLSEL information..... 83
7.2 Overview..... 83
7.3 Functional description..... 84

7.4 Application information..... 85
7.5 Memory map and register definition..... 86

Chapter 8 Flash Memory Module (FMU)..... 90
8.1 Chip-specific FMU information..... 90
8.2 Overview.....91
8.3 Functional description.....92
8.4 External signals..... 108
8.5 Initialization.....108
8.6 Memory map and registers.....109
8.7 Glossary..... 121

Chapter 9 Flash Controller (FMC).....123
9.1 Chip-specific FMC information..... 123
9.2 Overview.....123
9.3 Functional description.....124
9.4 External signals..... 126
9.5 Initialization and application information.....126

Chapter 10 FlexSPI.....127
10.1 Chip-specific FlexSPI information.....127
10.2 Overview.....128
10.3 Functional description.....130
10.4 External signals..... 182
10.5 Initialization.....183
10.6 Application information..... 183
10.7 Memory map and register definition..... 201
10.8 AHB memory map definition.....274

Chapter 11 Performance Monitor (CMX_PERFMON)..... 275
11.1 Chip-specific CMX_PERFMON information..... 275
11.2 Overview.....276
11.3 Functional description.....277
11.4 External signals..... 278
11.5 Initialization.....278
11.6 Memory map and register definition..... 278

Chapter 12 Signal Multiplexing..... 284
12.1 Introduction.....284
12.2 Pinout..... 284

Chapter 13 ROM API..... 317
13.1 Overview.....317
13.2 Functional description.....317

Chapter 14 In-System Programming (ISP)..... 376
14.1 Overview.....376
14.2 Functional description.....376

14.3 ISP protocol....378
14.4 Bootloader packet types....379
14.5 Bootloader command set....386
14.6 Bootloader status error codes....407
14.7 UART ISP....414
14.8 I2C ISP....416
14.9 SPI ISP....418
14.10 USB ISP....422
14.11 CAN ISP....424

Chapter 15 Boot ROM....426
15.1 Overview....426
15.2 Functional description....426
15.3 Boot modes....433
15.4 External memory support....460
15.5 eFuse definitions....469

Chapter 16 Debug....470
16.1 Introduction....470
16.2 Test and debug port connectivity....471
16.3 Debug ROM table....474
16.4 Cortex-M33 core debug....474
16.5 In-system programming access port (ISP-AP)....475
16.6 CTI....475
16.7 Low-power debug....476
16.8 Debug authentication....477

Chapter 17 System Controller (SYSCON)....478
17.1 Chip-specific SYSCON information....478
17.2 Overview....478
17.3 Signals....479
17.4 Memory map and register definition....479

Chapter 18 PowerQuad DSP Coprocessor and Accelerator (PowerQuad)....639
18.1 Chip-specific PowerQuad information....639
18.2 Overview....639
18.3 Functional description....640
18.4 External signals....654
18.5 Initialization....654
18.6 Application information....655
18.7 Memory map and register definition....670

Chapter 19 SmartDMA Controller....695
19.1 Chip-specific SmartDMA Controller information....695
19.2 Overview....698
19.3 Functional description....699
19.4 Memory map....700

Chapter 20 Neural Processing Unit (NPU)....701

20.1 Chip-specific NPU information....701
20.2 Overview....701
20.3 Functional description....704
20.4 Memory map and register definitions....706

Chapter 21 CoolFlux BSP32 Subsystem....707
21.1 Chip-specific CoolFlux BSP32 information....707
21.2 Overview....707
21.3 Functional description....709
21.4 External signals....710
21.5 Initialization....710
21.6 Application information....710
21.7 Physical memory map....710
21.8 CoolFlux BSP32 register descriptions....711

Chapter 22 Inter-CPU Mailbox (MAILBOX)....723
22.1 Chip-specific MAILBOX information....723
22.2 Overview....723
22.3 Functional description....724
22.4 Signals....724
22.5 Memory map and register definition....724

Chapter 23 Enhanced Direct Memory Access (eDMA) Controller....731
23.1 Chip-specific eDMA information....731
23.2 Overview....740
23.3 Functional description....742
23.4 External signals....747
23.5 Initialization....747
23.6 Memory map/register definition....759

Chapter 24 Wakeup Unit (WUU)....831
24.1 Chip-specific WUU information....831
24.2 Overview....833
24.3 Functional description....835
24.4 External signals....836
24.5 Initialization....836
24.6 Application information....837
24.7 Memory map and register definition....837

Chapter 25 Event Generator (EVTG)....881
25.1 Chip-specific EVTG information....881
25.2 Overview....881
25.3 Functional description....882
25.4 External signals....887
25.5 Application information....887
25.6 Memory map and register definition....888

Chapter 26 Input Multiplexing (INPUTMUX)....903
26.1 Chip-specific INPUTMUX information....903

26.2 Overview....903
26.3 Functional description....904
26.4 External signals....904
26.5 Memory map and register definition....904

Chapter 27 Semaphores2 (SEMA42)....1124
27.1 Chip-specific SEMA42 information....1124
27.2 Overview....1124
27.3 Functional description....1125
27.4 Memory map/register definition....1128
27.5 External signals....1132
27.6 Initialization....1132

Chapter 28 Interrupt Monitor (INTM)....1133
28.1 Chip-specific INTM information....1133
28.2 Overview....1133
28.3 Functional description....1134
28.4 External signals....1134
28.5 Initialization....1134
28.6 INTM register descriptions....1134

Chapter 29 Error Injection Module (EIM)....1142
29.1 Chip-specific EIM information....1142
29.2 Overview....1143
29.3 Functional description....1145
29.4 Initialization....1145
29.6 EIM register descriptions....1145

Chapter 30 Error Recording Module (ERM)....1158
30.1 Chip-specific ERM information....1158
30.2 Overview....1159
30.3 Functional description....1160
30.4 Initialization....1161
30.5 ERM register descriptions....1162

Chapter 31 Programmable Logic Unit (PLU)....1179
31.1 Chip-specific PLU information....1179
31.2 Overview....1179
31.3 Functional description....1180
31.4 External signals....1182
31.5 Initialization....1182
31.6 Memory Map and register definition....1183

Chapter 32 Reset....1192
32.1 Overview....1192
32.2 Power reset sources....1192
32.3 External reset sources....1193
32.4 Internal reset sources....1194
32.5 Reset type....1195

Chapter 33 Clocking..... 1198
33.1 Configuring the main clock and system clock.....1198
33.2 Clock generation.....1198
33.3 Module clocking.....1205
33.4 Set up PLL0.....1216
33.5 Set up PLL1.....1216

Chapter 34 System Clock Generator (SCG).....1217
34.1 Chip-specific SCG information..... 1217
34.2 Overview.....1218
34.3 Functional description.....1220
34.4 External signals..... 1238
34.5 Initialization.....1239
34.6 Application information..... 1239
34.7 Memory map and register definition..... 1243

Chapter 35 Power Management.....1311
35.1 Introduction.....1311
35.2 Power domains.....1311
35.3 Power modes.....1315
35.4 Module operation in low-power modes.....1317
35.5 Power supply configurations.....1322
35.6 Power optimization..... 1324

Chapter 36 VBAT.....1327
36.1 Chip-specific VBAT information.....1327
36.2 Overview.....1328
36.3 Functional description.....1329
36.4 External signals..... 1331
36.5 Initialization.....1332
36.6 Application information..... 1332
36.7 Memory map and register definition..... 1333

Chapter 37 System Power Control (SPC).....1373
37.1 Chip-specific SPC information.....1373
37.2 Overview.....1376
37.3 Functional description.....1377
37.4 External signals..... 1384
37.5 Initialization.....1384
37.6 Application information..... 1384
37.7 SPC register descriptions..... 1385

Chapter 38 Core Mode Controller (CMC)..... 1427
38.1 Chip-specific CMC information..... 1427
38.2 Overview.....1428
38.3 Functional description.....1428
38.4 External signals..... 1436
38.5 Initialization.....1436
38.6 Application information..... 1436
38.7 Memory map and register descriptions..... 1437

Chapter 39 Security overview....1470
39.1 Disclaimer....1470
39.2 Overview....1470
39.3 Immutable Root of Trust....1471
39.4 Life-cycle management....1471
39.5 Secure boot....1472
39.6 Secure update....1472
39.7 Secure debug....1472
39.8 IP protection....1472
39.9 Secure isolation....1472
39.10 Secure attestation....1472
39.11 Secure storage....1473
39.12 Trust provisioning....1473
39.13 Secure key management....1473
39.14 Anomaly Detection and Reaction....1473

Chapter 40 Cyclic Redundancy Check (CRC)....1474
40.1 Chip-specific CRC information....1474
40.2 Overview....1474
40.3 Functional description....1475
40.4 External signals....1477
40.5 Initialization....1477
40.6 Use cases....1478
40.7 Memory map and register descriptions....1480

Chapter 41 Analog-to-Digital Converter (ADC)....1485
41.1 Chip-specific ADC information....1485
41.2 Overview....1487
41.3 Functional description....1489
41.4 External signals....1500
41.5 Initialization....1500
41.6 ADC register descriptions....1502

Chapter 42 12-bit Digital-to-Analog Converter (DAC)....1553
42.1 Chip-specific 12-bit DAC information....1553
42.2 Overview....1554
42.3 Functional description....1555
42.4 DAC external signal descriptions....1557
42.5 Initialization....1557
42.6 Application Information....1557
42.7 Memory map and register definition....1558

Chapter 43 14-bit Digital-to-Analog Converter (DAC)....1574
43.1 Chip-specific 14-bit DAC information....1574
43.2 Overview....1575
43.3 Functional description....1576
43.4 DAC external signal descriptions....1577
43.5 Initialization....1578
43.6 Application Information....1578
43.7 Memory map and register definition....1578

Chapter 44 Low-Power Comparator (CMP)....1594
44.1 Chip-specific CMP information....1594
44.2 Overview....1596
44.3 Functional Description....1598
44.4 External signal descriptions....1611
44.5 Initialization....1611
44.6 Application information....1612
44.7 LPCMP register descriptions....1613

Chapter 45 Voltage Reference (VREF)....1635
45.1 Chip-specific VREF information....1635
45.2 Overview....1635
45.3 Functional description....1636
45.4 External signals....1638
45.5 Initialization....1639
45.6 Register descriptions....1639

Chapter 46 Operational Amplifier (OPAMP)....1645
46.1 Chip-specific OPAMP information....1645
46.2 Overview....1646
46.3 Functional description....1647
46.4 External Signals....1650
46.5 Initialization....1650
46.6 Application Information....1650
46.7 Memory map and register definition....1651

Chapter 47 SINC Filter....1658
47.1 Chip-specific SINC information....1658
47.2 Overview....1661
47.3 Functional description....1663
47.4 External signals....1707
47.5 Initialization....1708
47.6 Application information....1709
47.7 SINC register descriptions....1724

Chapter 48 Standard Counter/Timers (CTIMER)....1769
48.1 Chip-specific CTIMER information....1769
48.2 Overview....1771
48.3 Functional description....1773
48.4 External signals....1775
48.5 Initialization....1776
48.6 Application information....1776
48.7 CTIMER register descriptions....1776

Chapter 49 SCTIMER....1796
49.1 Chip-specific SCTIMER information....1796
49.2 Overview....1796
49.3 Functional description....1799
49.4 External signals....1804
49.5 Initialization....1804

49.6 Application information..... 1805
49.7 Memory map and register definition..... 1808

Chapter 50 Multi-Rate Timer (MRT)..... 1857
50.1 Chip-specific MRT information..... 1857
50.2 Overview..... 1857
50.3 Functional description..... 1858
50.4 External signals..... 1860
50.5 Initialization..... 1860
50.6 Memory map and register descriptions..... 1860

Chapter 51 Windowed Watchdog Timer (WWDT)..... 1869
51.1 Chip-specific WWDT information..... 1869
51.2 Overview..... 1869
51.3 Functional description..... 1871
51.4 External signals..... 1873
51.5 Initialization..... 1874
51.6 Memory map and register definition..... 1874

Chapter 52 Micro-Tick Timer (UTICK)..... 1882
52.1 Chip-specific UTICK information..... 1882
52.2 Overview..... 1882
52.3 Functional description..... 1883
52.4 External signals..... 1884
52.5 Initialization..... 1884
52.6 UTICK register descriptions..... 1884

Chapter 53 OS Event Timer (OSTIMER)..... 1891
53.1 Chip-specific OSTIMER information..... 1891
53.2 Overview..... 1891
53.3 Functional description..... 1892
53.4 External signals..... 1893
53.5 Initialization..... 1893
53.6 Memory map and register definition..... 1893

Chapter 54 Enhanced Flex Pulse Width Modulator (PWM)..... 1901
54.1 Chip-specific PWM information..... 1901
54.2 Overview..... 1901
54.3 Functional description..... 1903
54.4 External Signals..... 1935
54.5 PWM register descriptions..... 1936

Chapter 55 Quadrature Decoder (QDC)..... 2062
55.1 Chip-specific QDC information..... 2062
55.2 Overview..... 2062
55.3 Functional description..... 2063
55.4 External signals..... 2076
55.5 Memory map and registers..... 2077

Chapter 56 Real-Time Clock Subsystem (RTC_SUBSYSTEM)..... 2104
56.1 Chip-specific RTC_SUBSYSTEM information..... 2104
56.2 Overview..... 2104
56.3 Functional description..... 2105
56.4 External signals..... 2106
56.5 Initialization..... 2106
56.6 RTC_SUBSYSTEM register descriptions..... 2106

Chapter 57 Real-Time Clock (RTC)..... 2111
57.1 Chip-specific RTC information..... 2111
57.2 Overview..... 2111
57.3 Functional description..... 2112
57.4 External Signals..... 2117
57.5 Memory map and registers..... 2117

Chapter 58 Frequency Measurement (FREQME)..... 2140
58.1 Chip-specific Frequency Measurement information..... 2140
58.2 Overview..... 2140
58.3 Functional description..... 2140
58.4 External signals..... 2142
58.5 Initialization..... 2142
58.6 Memory map and register definition..... 2142

Chapter 59 Low-Power Timer (LPTMR)..... 2151
59.1 Chip-specific LPTMR information..... 2151
59.2 Overview..... 2152
59.3 Functional description..... 2152
59.4 External signals..... 2155
59.5 Initialization..... 2155
59.6 Application information..... 2155
59.7 Memory map and register definition..... 2156

Chapter 60 External Watchdog Monitor (EWM)..... 2163
60.1 Chip-specific EWM information..... 2163
60.2 Overview..... 2163
60.3 Functional Description..... 2164
60.4 External signals..... 2167
60.5 Memory Map..... 2168

Chapter 61 USB Full-Speed Host and Device Controller (USBFS)..... 2175
61.1 Chip-specific USBFS information..... 2175
61.2 Overview..... 2175
61.3 Functional description..... 2178
61.4 External signals..... 2190
61.5 Initialization..... 2191
61.6 Application information..... 2191
61.7 Memory map and register definitions..... 2197

Chapter 62 Universal Serial Bus 2.0 High-Speed Integrated PHY (USBHS_PHY)..... 2250
62.1 Chip-specific USBHS_PHY information..... 2250
62.2 Overview..... 2250
62.3 Functional description..... 2251
62.4 External signals..... 2258
62.5 Initialization..... 2258
62.6 Application information..... 2259
62.7 USBPHY register descriptions..... 2260
62.8 Register macro usage..... 2310
62.9 References..... 2310

Chapter 63 USB Device Charger Detection Module (USBDCD)..... 2311
63.1 Chip-specific USBDCD information..... 2311
63.2 Overview..... 2311
63.3 Functional description..... 2312
63.4 External signals..... 2325
63.5 Initialization..... 2326
63.6 Application information..... 2326
63.7 USBDCD register descriptions..... 2327

Chapter 64 Universal Serial Bus High-Speed Controller (USBHS)..... 2338
64.1 Chip-specific USBHS information..... 2338
64.2 Overview..... 2338
64.3 Functional description..... 2339
64.4 External signals..... 2343
64.5 Initialization..... 2344
64.6 Application information..... 2345
64.7 USB non-core memory map/register definition..... 2375
64.8 USB core memory map/register definition..... 2380

Chapter 65 Ultra Secured Digital Host Controller (uSDHC)..... 2465
65.1 Chip-specific uSDHC information..... 2465
65.2 Overview..... 2465
65.3 Functional description..... 2468
65.4 External signals..... 2493
65.5 Application information..... 2494
65.6 uSDHC memory map and register definition..... 2511

Chapter 66 Low-Power Flexible Communications Interface (LP_FLEXCOMM) 2583
66.1 Chip-specific LP_FLEXCOMM information..... 2583
66.2 LP_FLEXCOMM..... 2583
66.3 Low-Power Inter-Integrated Circuit (LPI2C)..... 2590
66.4 Low-Power Serial Peripheral Interface (LPSPI)..... 2661
66.5 Low-Power Universal Asynchronous Receiver/Transmitter (LPUART)..... 2709

Chapter 67 Synchronous Audio Interface (SAI)..... 2780
67.1 Chip-specific SAI information..... 2780
67.2 Overview..... 2781

67.3 Functional description....2782
67.4 External signals....2789
67.5 Initialization....2790
67.6 Memory map and register definition....2790

Chapter 68 Flexible Controller Area Network (FlexCAN)....2830
68.1 Chip-specific CAN information....2830
68.2 Overview....2830
68.3 Functional description....2833
68.4 External signal descriptions....2883
68.5 Initialization and application information....2883
68.6 Memory map and register definition....2884

Chapter 69 EMVSIM....2980
69.1 Chip-specific EMVSIM information....2980
69.2 Overview....2980
69.3 Functional description....2982
69.4 External signals....2994
69.5 Initialization....2994
69.6 Application information....2996
69.7 Memory map and registers....2999

Chapter 70 Ethernet QoS (ENET)....3031
70.1 Chip-specific ENET information....3031
70.2 Overview....3031
70.3 Functional description....3033
70.4 Signals....3099
70.5 Memory Map and register definition....3100

Chapter 71 Flexible Input/Output (FlexIO)....3320
71.1 Chip-specific FlexIO information....3320
71.2 Overview....3324
71.3 Functional description....3326
71.4 External signals....3336
71.5 Initialization....3336
71.6 Application information....3336
71.7 Memory map and registers....3356

Chapter 72 Improved Inter-Integrated Circuit (I3C)....3403
72.1 Chip-specific I3C information....3403
72.2 Overview....3403
72.3 Functional description....3405
72.4 External signals....3418
72.5 Initialization....3418
72.6 Application information....3420
72.7 I3C register descriptions....3422

Chapter 73 General Purpose Input/Output (GPIO)....3524
73.1 Chip-specific GPIO information....3524

73.2 Overview....3525
73.3 Functional description....3526
73.4 External signals....3528
73.5 Initialization....3529
73.6 Application information....3529
73.7 Memory map and register definition....3529

Chapter 74 Pin Interrupt and Pattern Match (PINT)....3551
74.1 Chip-specific PINT information....3551
74.2 Overview....3552
74.3 Functional description....3552
74.4 External signals....3555
74.5 Initialization....3555
74.6 Application information....3555
74.7 Memory map and register definition....3558

Chapter 75 Port Control (PORT)....3573
75.1 Chip-specific PORT information....3573
75.2 Overview....3574
75.3 Functional description....3574
75.4 Initialization....3576
75.5 Application information....3576
75.6 Memory map and register definition....3577

Chapter 76 PDM Microphone Interface (MICFIL)....3687
76.1 Chip-specific MICFIL information....3687
76.2 Overview....3687
76.3 Functional description....3688
76.4 External signals....3697
76.5 Initialization....3697
76.6 Application information....3697
76.7 MICFIL register descriptions....3697

Chapter 77 Touch Sensing Input (TSI)....3716
77.1 Chip-specific TSI information....3716
77.2 Overview....3716
77.3 Functional description....3718
77.4 External signals....3727
77.5 Initialization....3728
77.6 Application information....3728
77.7 Register definition....3728

Appendix A Revision history....3760
A.1 Revision history....3760

Legal information....3761

Chapter 1
About This Manual

1.1 About This Document

1.1.1 Audience

This manual is intended for the board-level product designers and product software developers. This manual assumes that the reader has a background in computer engineering and/or software engineering and understands the concepts of the digital system design, microprocessor architecture, Input/Output (I/O) devices, industry standard communication, and device interface protocols.

1.1.2 Organization

This document is organized into two main sets of chapters.

1. First set covers the chip at a system level and provides an architectural overview. It also covers the system memory map, system-level interrupt events, clock, reset and system controller.
2. Chapters in the second set provides a technical description of individual modules along with chip-specific contents at the beginning of each chapter.

1.1.3 Suggested Reading

This section lists the additional resources that provide background for the information in this manual, as well as general information about the architecture.

1.1.3.1 General Information

The following documentation provides useful background information about the Arm® Cortex® processor.

For information about the Arm Cortex processor see:

- **http://infocenter.arm.com**

Refer Security Primitives: Requirements in (I)IoT Systems document to familiarize with NXP's security nomenclature.

1.1.3.2 Related Documentation

For a current list of documentation, refer to **http://www.nxp.com**.

1.1.4 Conventions

This document uses the following notational conventions:

cleared / set	When a bit has a value of zero, it is said to be cleared; when it has a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics indicate variable command parameters, for example, **bcctr***x*. The book titles in the text are set in italics.
15	An integer in decimal.
0x	the prefix to denote a hexadecimal number.
0b	The prefix to denote a binary number. Binary values of 0 and 1 are written without a prefix.
n'H4000CA00	The n-bit hexadecimal number.

BLK_REG_NAME	The register names are all uppercase. The block mnemonic is prepended with an underscore delimiter (_).
BLK_REG[FIELD]	The fields within registers appear in brackets. For example, ESR[RLS] refers to the Receive Last Slot field of the ESAI Status Register.
BLK_REG[*n*]	The bit number *n* within the BLK.REG register.
BLK_REG[*l:r*]	The register bit ranges. The ranges are indicated by the left-most bit number *l* and the right-most bit number *r*, separated by a colon (:). For example, ESR[15:0] refers to the lower half word in the ESAI Status Register.
x, U	In some contexts, such as signal encodings, an unitalicized x indicates a "don't care" or "uninitialized". The binary value can be 1 or 0.
x	An italicized *x* indicates an alphanumeric variable.
n, m	Italicized *n or m* represent integer variables.
!	Binary logic operator NOT.
&&	Binary logic operator AND.
||	Binary logic operator OR.
^ or <O+>	Binary logic operator XOR. For example, A <O+> B.
|	Bit-wise OR. For example, 0b0001 | 0b1000 yields the value of 0b1001.
&	Bit-wise AND. For example, 0b0001 and 0b1000 yields the value of 0b0000.
{A,B}	Concatenation, where the *n*-bit value A is prepended to the *m*-bit value B to form an (*n*+*m*)-bit value. For example, {0, REG*m* [14:0]} yields a 16-bit value with 0 in the most significant bit.
- or grey fill	Indicates a reserved bit field in a register. Although these bits can be written to with ones or zeros, they always read zeros.
>>	Shift right logical one position.
<<	Shift left logical one position.
<=	Assignment.
==	Compare equal.
!=	Compare not equal.
>	Greater than.
<	Less than.

1.1.5 Register Access

1.1.5.1 Register Diagram Field Access Type Legend

This figure provides the interpretation of the notation used in the register diagrams for a number of common field access types:

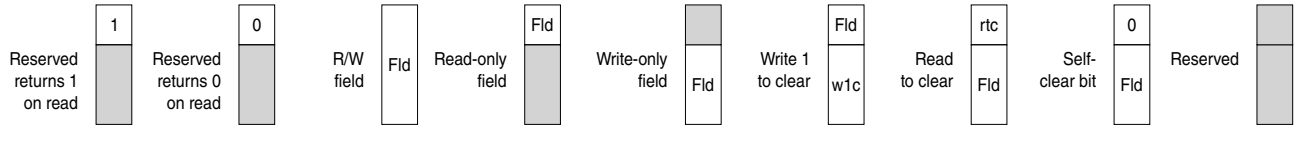


Figure 1. Register Field Conventions

NOTE

For reserved register fields, the software should mask off the data in the field after a read (the software can't rely on the contents of data read from a reserved field) and always write all zeros.

1.1.5.2 Register Macro Usage

A common operation is to update one field without disturbing the contents of the remaining fields in the register. Normally, this requires a read-modify-write (RMW) operation, where the CPU reads the register, modifies the target field, then writes the results back to the register. This is an expensive operation in terms of CPU cycles, because of the initial register read.

To address this issue, some hardware registers are implemented as a group, including registers that can be used to either set, clear, or toggle (SCT) individual bits of the primary register. When writing to an SCT register, all the bits set to 1 perform the associated operation on the primary register, while the bits set to 0 are not affected. The SCT registers always read back 0 and should be considered write-only. The SCT registers are not implemented if the primary register is read-only.

With this architecture, it is possible to update one or more fields using only register writes. First, all bits of the target fields are cleared by a write to the associated clear register, then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one-bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (that is, one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read-modify-write operations. When an atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

A set of SCT registers is offered for registers in many modules on this device, as described in this manual.
In a module memory map table, the suffix _SET, _CLR, or _TOG is added to the base name of the register.
For example, the CCM_ANALOG_PLL_ARM register has three other registers called CCM_ANALOG_PLL_ARM_SET, CCM_ANALOG_PLL_ARM_CLR, and CCM_ANALOG_PLL_ARM_TOG.

In the sub-section that describes one of these sets of registers, a short-hand convention is used to denote that a register has the SCT register set. There is an italicized *n* appended to the end of the short register name. Using the above example, the name used for this register is CCM_ANALOG_PLL_ARM*n*. When you see this designation, there is a SCT register set associated with the register, and you can verify this by checking it in the memory map table. The address offset for each of these registers is given in the form of the following example:

Address: 20C_8000h base + 0h offset + (4d × i), where i=0d to 3d

In this example, the address for each of the base registers and their three SCT registers can be calculated as:

Register	Address
CCM_ANALOG_PLL_ARM	20C_8000h
CCM_ANALOG_PLL_ARM_SET	20C_8004h
CCM_ANALOG_PLL_ARM_CLR	20c_8008h
CCM_ANALOG_PLL_ARM_TOG	20C_800Ch

Chapter 2
Introduction

2.1 Overview

The MCX N dual-core microcontroller series combines two Arm Cortex-M33 cores with a Neural Processing Unit (NPU) Neutron1-16, a CoolFlux BSP32, a PowerQuad DSP Co-processor, TrustZone capabilities, and multiple high-speed connectivity options running at 150 MHz. To support a wide variety of applications, the MCX N series includes advanced serial peripherals, timers, high-precision analog, and state-of-the-art security features like secure user code, data, and communications. All MCX N dual-core products include dual-bank flash which supports read while write operation from internal flash. They also support large external serial memory configurations. The MCX N dual-cores series are as follows:

- **N52:** Mainstream MCU with a second M33 core, CoolFlux BSP32 programmable for PLC connectivity, PowerQuad, advanced timers, analog, high-speed connectivity (including Hi-Speed USB), and FlexIO which can be programmed as an LCD controller.
- **N53:** Mainstream MCU with a second M33 core, Edge AI-ready NPU, PowerQuad, advanced timers, analog, high-speed connectivity (including Hi-Speed USB), audio, and FlexIO which can be programmed as an LCD controller.
- **N54**: Mainstream MCU with a second M33 core, Edge AI-ready NPU, PowerQuad, advanced timers, analog, high-speed connectivity (including Hi-Speed USB), 10/100 Ethernet, and FlexIO which can be programmed as an LCD controller.
- **N94**: Integration CPU, NPU, DSP, advanced timers, high precision analog, high-speed connectivity including Hi-Speed USB, CAN 2.0, 10/100 Ethernet, and FlexIO which can be programmed as an LCD controller.

The MCX N single-core series microcontrollers combine the Arm Cortex-M33 TrustZone® core with multiple high-speed connectivity options running at 150 MHz. To support a wide variety of applications, the MCX N series includes advanced serial peripherals, timers, high-precision analog, and state-of-the-art security features like secure user code, data, and communications. All MCX N products include dual-bank flash which supports read while write operation from internal flash. The MCX N single-core series are general purpose MCUs with advanced peripherals and a range of serial, analog, and high-speed connectivity. The MCX N single-core series are as follows:

- **N24:** General purpose MCU with advanced peripherals and a range of serial, analog and high-speed connectivity including 10/100Mbps Ethernet. It integrates up to 2MB of flash memory and 512kB of SRAM alongside motor control and graphics capabilities.

The chip includes these key features:

Table 1. Key features

Function	Features
Security	• TrustZone for Armv8M • Secure boot, firmware update, and debug authentication using ROM • EdgeLock® secure subsystem (ELS) • Public-key cryptography (PKC) • External and internal flash memory interface with on-the-fly PRINCE decryption and encryption • Physically Unclonable Function (PUF) hardware options • Factory Root of Trust (RoT) programming • Tamper detection • Analog and digital glitch detection

Table continues on the next page...

Table 1. Key features (continued)

Function	Features
	• Code Watchdog (CDOG) • Intrusion and Tamper Response Controller (ITRC)
Industrial strength	• Industrial temperature rating • Industrial communication protocol support (CAN-FD) • High-resolution mixed signal analog • BLDC and PMSM motor control support • Integrated sensor interfaces: I3C, I2C, SPI, and UART • 15-year longevity • Configurable RAM ECC • Ethernet (ENET) with QoS
Power-efficient operating modes	• Down to 57 µA/MHz (3.3 V, @25 ℃) in OD Active Mode (while(1) executing from flash, DC/DC enabled) • 170 µA in Deep Sleep mode (full 512 KB SRAM retention, 3.3 V, @25 ℃) • 5.2 µA in Power Down mode (full 512 KB SRAM retention, 3.3 V, @25 ℃) • Down to 2.0 µA in Deep Power Down mode, 5.3 ms wake-up (RTC enabled 8 KB RAM and Reset pin enabled, @25 ℃)

2.2 Target applications

- **Industrial**
  - Energy Storage and Management System
  - Smart Metering
  - Power Line Communication
  - Factory Automation
  - Industrial HMI
  - Mobile Robotics Ecosystem
  - Motion Control and Robotics
  - Motor Drives
  - Brushless DC Motor (BLDC) Control Permanent Magnet Synchronous Motor (PMSM)
  - Edge AI/ML Anomaly Detection and Predictive Maintainance
- **Smart Home**
  - Home Control Panel
  - Home Security and Surveillance
  - Home Security and Surveillance
  - Major Home Appliances
  - Robotic Appliance

- Smart Speaker
- Soundbar
- Gaming Accessories
- Smart Lighting
- Smart Power Socket and Light Switch
- Edge AI/ML Vision and Voice Detection / Recognition

2.3 Block diagram

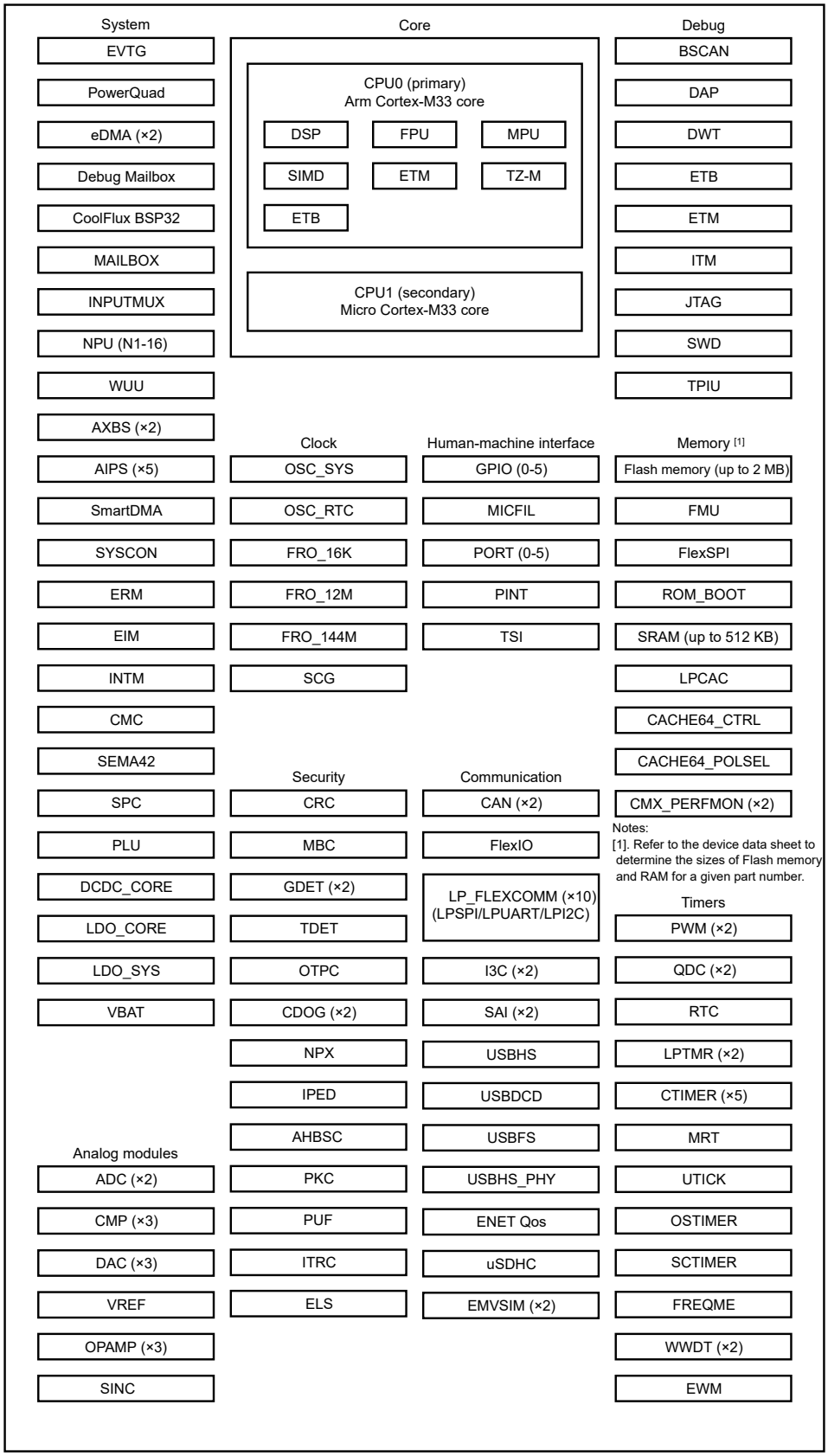


Figure 2. Features block diagram

NOTE

See the datasheet for information on peripheral availability and quantity for the specific part number and package.

Figure 3 shows the chip's block diagram, including bus connections.

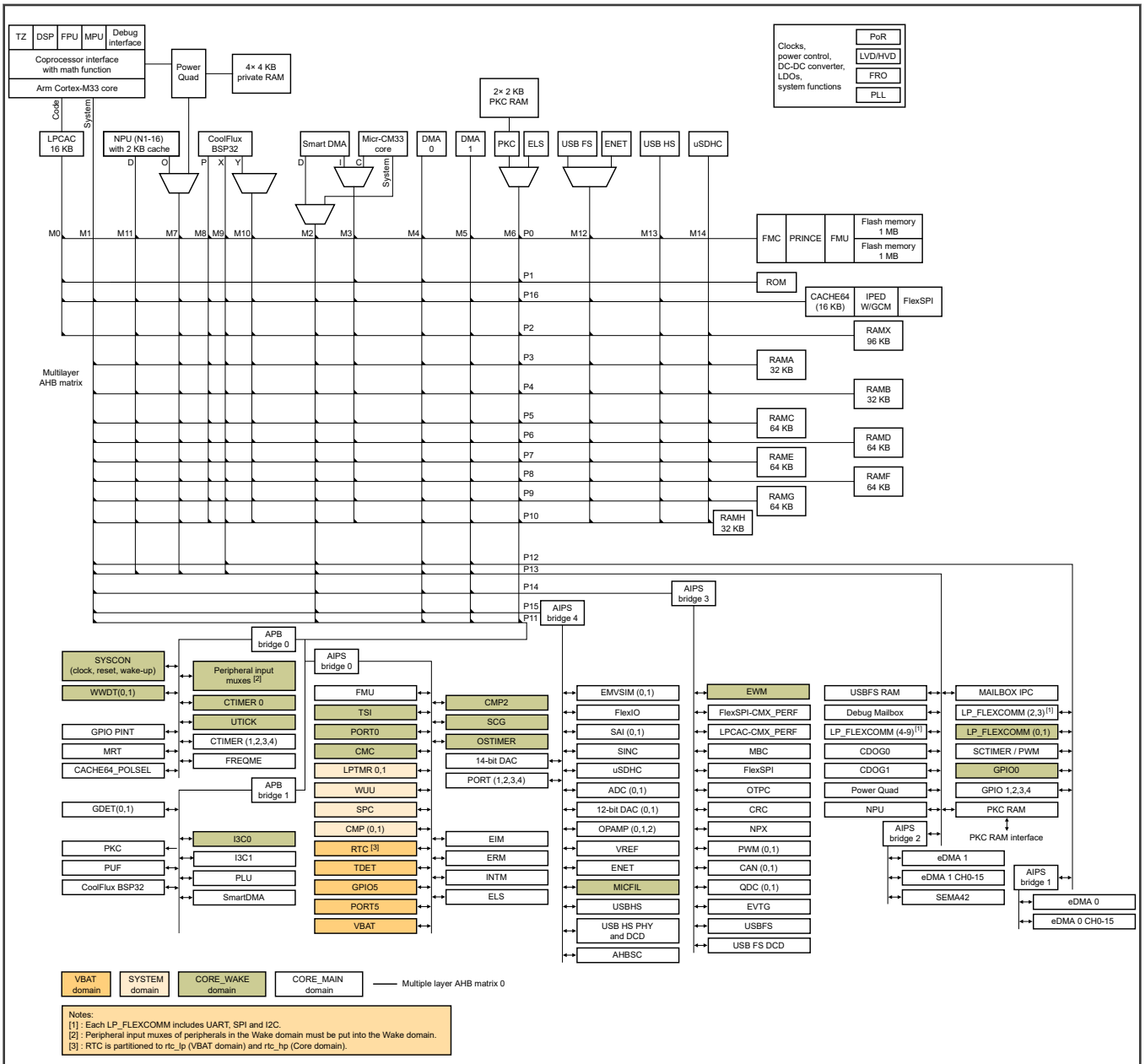


Figure 3. Bus matrix block diagram

NOTE

See the datasheet for information on peripheral availability and quantity for the specific part number and package.

2.4 System bus priority and arbitration

As shown in Figure 3 , this device includes a number of masters that can access the various slave ports. The multilayer AHB matrix allows for concurrent accesses when multiple masters are attempting to access different slave ports. If multiple masters attempt an access to the same slave port at the same time then arbitration is required.

The SYSCON_AHBMATPRIO register is where the programmable priorities for each of the master ports can be configured. Masters are assigned a priority value between zero and three with three being the highest priority. If two ports have the same priority, then the lowest port number is given priority.

There are some master ports that are shared between two masters. Where the port is shared, only one of the masters can have an active access at a time. The priority between two masters sharing a port uses a fixed arbitration scheme. The table below lists the master or masters for each of the ports. In the case where the port is shared, the high-priority master is specified:

Table 2. AHB bus matrix ports

Master port	Master	Accessible slave ports	Inaccessible slave ports	AHB ID
M0	CPU0 (CM33) code bus	P0 - P2 and P16	P3 - P15	0
M1	CPU0 (CM33) system bus	P3 - P16	P0 - P2	1
M2	CPU1 (Micr-CM33) code bus (low-priority)	P0 and P2 - P16	P1	2
	SmartDMA data bus (high-priority)			3
M3	CPU1 (Micr-CM33) system bus (low-priority)	P0 - P10 and P16	P11 - P15	4
	SmartDMA instruction bus (high-priority)			5
M4	DMA0	P0 - P16	None	6
M5	DMA1	P0 and P2 - P16	P1	7
M6	PKC (low-priority)	P0 - P11 and P16	P12 - P15	8
	ELS (high-priority)			9
M7	PowerQuad (low-priority)	P0, P2 - P10, P13, and P16	P1, P11 - P12, and P14 - P15	10
	NPU operand bus (high-priority)			11
M8	CoolFlux BSP32 program memory bus	P0, P2 - P10, and P16	P1 and P11 - P15	12
M9	CoolFlux BSP32 X-data memory bus	P0, P2 - P10, P12 - P13, and P16	P1, P11, and P14 - P15	13
M10	CoolFlux BSP32 Y-data memory bus	P0, P2 - P10, and P16	P1 and P11 - P15	14
M11	NPU data bus	P0, P2 - P10, P13, and P16	P1, P11 - P12, and P14 - P15	16
M12	USBFS (low-priority)	P0, P2 - P10, and P16	P1 and P11 - P15	17
	Ethernet (high-priority)			18
M13	USBHS	P0, P2 - P10, and P16	P1 and P11 - P15	19
M14	uSDHC	P2 - P10	P0 - P1 and P11 - P16	20

2.5 Functional overview

Table 3 shows the chip modules organized by functional category.

Table 3. Module functional categories

Module category	Description
Core	• The Arm Cortex®-M33 is part of the Cortex-M Series of processors targeting microcontroller cores focused on cost-sensitive, deterministic, and interrupt-driven environments. • The Cortex-M33 processor is based on the Armv8-M Architecture and ThumbR-2 ISA and is upward compatible with the Cortex-M7, M4, M3, M1, M0, and M0+.
System	• PowerQuad DSP Coprocessor and Accelerator (PowerQuad) • Debug Mailbox • SmartDMA Controller • Neural Processing Unit (NPU N1-16) • CoolFlux BSP32 • Inter-CPU Mailbox (MAILBOX) • Enhanced Direct Memory Access (eDMA) • Wake-Up Unit (WUU) • Event Generator (EVTG) • AHB Cross-Bar Switch (AXBS) Lite • Peripheral Inputmux • Semaphore (SEMA42) • Interrupt Monitor (INTM) • Error Injection Module (EIM) • Error Reporting Module (ERM) • Programmable Logic Unit (PLU) • System Power Controller (SPC) • Core Mode Controller (CMC)
Memory	• ROM-BOOT and ROM-CODE • Static Random Access Memory (SRAM) • AHB Low Power Cache Controller (LPCAC) • CACHE64 Controller (CACHE64_CTRL) • CACHE64 Policy Select (CACHE64_POLSEL) • Flash Management Unit (FMU) • Flash Memory Controller (FMC) supporting counter mode PRINCE • Flexible Serial Peripheral Interface (FlexSPI) supporting IPED with GCM. FlexSPI also supports Octal and Quad memory interfaces. • Performance Monitor (CMX_PERFMON)

Table continues on the next page...

Table 3. Module functional categories (continued)

Module category	Description
Clock	• VBAT — Crystal Oscillator - Real Time Clock (OSC_RTC) — 16 K Free Running Oscillator (FRO_16K) • System Clock Generator (SCG) — Crystal Oscillator - System (OSC_SYS) — 144 M Free Running Oscillator (FRO_144M) — 12 M Free Running Oscillator (FRO_12M) — PLL and USB PLL
Security	• EdgeLock Secure Subsystem (ELS) • Public Key Cryptography (PKC) • Physical Unclonable Function (PUF) • Cyclic Redundancy Check (CRC) • PRINCE encryption/decryption • Digital tamper (TDET) • Code Watchdog (CDOG) • Digital and analog Glitch Detect (GDET) • OTP Controller (OTPC) • Intrusion and Tamper Response Controller (ITRC) • AHB Secure Controller (AHBSC) • Memory Block Checker (MBC) • TrustZone
Analog	• 16/12-bit Analog-to-Digital Converter (ADC) • 14-bit Digital-to-Analog Converter (DAC) • 12-bit Digital-to-Analog Converter • Low Power Comparator (LPCMP) • Voltage Reference (VREF) • Operational Amplifier (OPAMP) • Sigma-Delta Modulator (SINC)
Timer	• Standard Counter/Timer (CTIMER) • SCTimer • Multi Rate Timer (MRT) • Windowed Watchdog Timer (WWDT) • Micro-Tick (UTICK) Timer

Table continues on the next page...

Table 3. Module functional categories (continued)

Module category	Description
	• OS Event Timer (OSTIMER) • Enhanced Flex Pulse Width Modulator (PWM) • Quadrature Decoder (QDC) • Real Time Clock (RTC) • Frequency Measurement (FREQME) • Low-power Timer (LPTMR) • External Watchdog Monitor (EWM)
Communication	• Universal Serial Bus - Full Speed (USBFS) and Transceiver • USB High-Speed (USBHS) Controller and Physical Layer Interface (PHY) • Universal Serial Bus - Device Charger Detect (USBDCD) • Ultra Secured Digital Host Controller (uSDHC) • LP_FLEXCOMM with Low Power Inter-Integrated Circuit (LPI2C), Low Power Serial Peripheral Interface (LPSPI), and Low Power Universal Asynchronous Receive/Transmit (LPUART) support • Serial Audio Interface (SAI) • FlexCAN with Flexible Data rate (FlexCAN FD) • Euro, MasterCard, and Visa Subscriber Identification Module (EMVSIM) • Ethernet QoS (ENET) • Flexible Input/Output (FlexIO) • Improved Inter-Integrated Circuit (I3C)
Human Machine Interface (HMI)	• General Purpose Input/Output (GPIO) • Port Control (PORT) • Pin Interrupt and Pattern Match (PINT) • PDM Microphone Interface (MICFIL) • Touch Sense Interface (TSI)

2.5.1 Core

The following core modules are available on this chip.

Table 4. Core modules

Module	Description
CPU0	An Arm Cortex-M33 processor that runs at a frequency of up to 150 MHz (device revision 1B only). The configuration of this instance includes MPU, FPU, DSP, ETM, TrustZone, and coprocessor interface.

Table continues on the next page...

Table 4. Core modules (continued)

Module	Description
CPU1	An Arm Cortex-M33 coprocessor that runs at a frequency of up to 150 MHz (device revision 1B only). The configuration of this instance supports system tick timer but does not include MPU, FPU, DSP, ETM, and TrustZone.
Digital Signal Processing Extensions (DSP)	The Cortex-M33 processor features extended single-cycle Multiply Accumulate (MAC) instructions, optimized Single Instruction Multiple Data (SIMD) arithmetic, and saturating arithmetic instructions.
Floating Point Unit (FPU)	Provides single-precision floating point computation, compliant to the IEEE Standard for Floating-Point Arithmetic (IEEE 754).
Memory Protection Unit (MPU)	Provides support for eight unified protection regions, overlapping protection regions with ascending region priority, and access permissions. MPU mismatches and permission violations invoke the HardFault handler.
Nested Vectored Interrupt Controller (NVIC)	The Armv8M exception model and Nested-Vectored Interrupt Controller (NVIC) implement a relocatable vector table supporting external interrupts, a single non-maskable interrupt (NMI), and priority levels.
System Tick Timer (SysTick)	See the *Armv8M Architecture Reference Manual* for more information about this system timer.

2.5.2 Debug

The following debug modules are available on this chip.

Table 5. Debug modules

Module	Description
Boundary Scan (BSCAN)	Allows external circuitry testing of a chip. The serial scan chain also permits the system signals flowing into and out of the system logic to be sampled and examined without causing interference with the normal device operation.
Cross Trigger Interface (CTI)	Enables the debug logic of the chip to interact with each other (cross-trigger).
Data Watchpoint and Trace (DWT)	A generic name for modules that allow debug access of the Cortex-M33. The DWT consists of the Debug Watchpoint and Trace (DWT) module and the Flash Patch and Breakpoint (FPB) unit.
Debug Access Port (DAP)	Enables real-time access to the chip registers from an external debugger without halting the processor cores.
Embedded Trace Buffer (ETB)	On-chip memory used to store real-time trace instruction and data trace information from the Embedded Trace Macrocell (ETM).
Embedded Trace Macrocell (ETM)	Real-time trace module providing real-time instruction and data tracing of a processor core.

Table continues on the next page...

Table 5. Debug modules (continued)

Module	Description
Instruction Trace Macrocell (ITM)	Provides a memory-mapped register interface that applications can use to write logging or event words for profiling software.
Joint Test Action Group (JTAG)	Implements serial communication protocol for communicating with the Test Access Port (TAP).
Serial Wire Debug (SWD)	A serial communication interface used for debugging devices with multiple cores while only requiring a single external interface.
Trace Port Interface Unit (TPIU)	Acts as a bridge between the on-chip trace data from the modules (such as ETM or ITM which have separate system IDs) to the external world.

2.5.3 System

The following system modules are available on this chip.

Table 6. System modules

Module	Description
Power Quad	Hardware accelerator targeting common calculations in DSP applications.
Debug mailbox	Supports Arm Serial Wire Debug mode.
SmartDMA	Supports unique reduced instruction sets and performs event- and IO-driven handling to offload the work from the Arm processor.
CoolFlux BSP32	Dual-Harvard, dual-multiplier processor based on the ultra-low power CoolFlux architecture. Supports 16-bit and 32-bit scalar operations, as well as complex SIMD operations and data types.
Inter-CPU Mailbox (MAILBOX)	Provides simple communication and resource allocation between CPUs.
Event Generator (EVTG)	Supports the generation of a configurable number of EVENT signals. EVTG includes two AND/OR/INVERT (AOI) modules and one configurable flip-flop.
AHB Cross-Bar Switch (AXBS) Lite	Connects bus masters and bus slaves. This allows the bus masters to access different bus slaves simultaneously and provides arbitration among the bus masters when they access the same slave.
Peripheral Bridge(AIPS_Lite)	Converts an AMBA AHB interface to a peripheral interface that allows the Peripheral Bridge2 (PBRIDGE2) controller to interface to multiple peripherals. This device has five peripheral bridges.
System Controller (SYSCON)	Provides controls and configurations of the system and peripherals for the multiple functions.

Table continues on the next page...

Table 6. System modules (continued)

Module	Description
Core Mode Controller (CMC)	Provides control and protection on entry and exit to each power mode, control for the System Power Controller (SPC), and reset entry and exit for the complete device.
Enhanced Direct Memory Access (eDMA)	Performs source and destination address calculations and data-movement operations. Capable of performing complex data transfers with minimal intervention from a host processor.
NPU	A Neural Network processing accelerator. It is a configurable architecture comprising weight compression, compute engine, and data canvas.
Wake-up Unit (WUU)	Allows selection of external pins and internal modules as interrupt wake-up sources from Power Down and Deep Power Down modes.
Peripheral Input Multiplexing (INPUTMUX)	Allows the trigger output of one peripheral to be connected to the trigger input of a second peripheral.
Semaphore (SEMA42)	A memory-mapped module that provides the robust hardware support needed in multi-core systems for implementing semaphores. It provides a simple mechanism to achieve lock and unlock operations via a single write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.
Interrupt monitor (INTM)	Provides a mechanism to monitor the latency of the responses on interrupt requests.
Error Injection Module (EIM)	Provides a method for diagnostic coverage of internal memories (for example, system RAM, cache RAMs, and peripheral memories).
Error Reporting Module (ERM)	Provides information and optional interrupt notification on memory error events associated with error correction code (ECC) and parity.
System Power Controller (SPC)	Provides control over the operation and configuration of the system power generation modules to optimize power consumption for the level of functionality needed.
Direct Current/Direct Current Converter - Core (DCDC_CORE)	A voltage converter for generating the core voltage for the chip.
Low Drop Out Regulator - Core (LDO_CORE)	A voltage regulator for generating the core voltage for the chip.
Low Drop Out Regulator - System (LDO_SYS)	A voltage regulator for generating the system voltage for the chip.
Programmable Logic Unit (PLU)	Monitors the state of input devices and makes decisions based on a custom program to control the state of output devices.

2.5.4 Memory

The following memory modules are available on this chip.

Table 7. Memory modules

Module	Description
Static Random Access Memory (SRAM)	Internal system SRAM memory. Each individual block of SRAM can be configured to be retained in low-power modes.
AHB Low Power Cache Controller (LPCAC)	A processor-local level 1 (L1) bus cache controller for use with cores using AMBA-AHB input/output buses.
CACHE64 Controller (CACHE64_CTRL)	Includes cache memory controller (CACHE64), which is a general purpose AMBA AHB bus protocol cache.
CACHE64 Policy Select (CACHE64_POLSEL)	Monitors incoming AHB addresses to define the policy that the cache controller (CACHE64_CTRL) uses.
Flash Management Unit (FMU)	Manages the interface between the chip and the on-chip flash memory.
Flash Memory Controller (FMC)	A programmable, non-volatile flash memory that can store executable program code or data.
Flexible Serial Peripheral Interface (FlexSPI)	Interfaces with external serial flash devices.
Performance Monitor (CMX_PERFMON)	Contains counters which can be configured to count events in order to calculate performance of a CPU, cache, or memory.

2.5.5 Clock

The following clock modules are available on this chip.

Table 8. Clock sources

Module	Description
Crystal Oscillator - Real Time Clock (OSC_RTC)	Generates, in conjunction with an external 32 kHz crystal, a 32 kHz reference clock for the RTC that can also be used by the chip.
Free Running Oscillator - 16 K (FRO_16K)	An ultra low-power internal 16.384 kHz clock source. It is functional down to VBAT mode. The FRO_16K is trimmed to +/- 6% accuracy over the entire voltage and temperature range.
System Clock Generator (SCG)	Provides you with access to the configuration control registers for the system level clock sources.
Crystal Oscillator - System (OSC_SYS)	Generates, in conjunction with an external crystal or resonator, a reference clock for the chip.
Free Running Oscillator - 144 MHz (FRO_144M)	An internal clock source that generates a 144 MHz reference clock for the RTC that can also be used by the chip.
Free Running Oscillator - 12 MHz (FRO_12M)	An internal clock source that generates a 12 MHz frequency for use by the chip.

Table continues on the next page...

Table 8. Clock sources (continued)

Module	Description
PLL and USB PLL	PLL0 and PLL1 allow the chip operation up to the maximum CPU rate without the need for a high-frequency external clock. PLL0 and PLL1 can run from the internal FRO_144M, the 40 MHz OSC, or the 32.768 kHz RTC oscillator. Both support fractional division and spread-spectrum. There is a PLL in USBHS PHY to provide 480 MHz clock for USBHS. USB PLL uses external 16 MHz, 19.2 MHz, 20 MHz, 24 MHz, or 32 MHz as a reference clock. The USB PLL clock can be used as a system clock source.

2.5.6 Security

The following security modules are available on this chip.

Table 9. Security modules

Module	Description
ELS	Security subsystem ensuring key isolation.
PKC	Public-key cryptography accelerator
Physically Unclonable Function (PUF) SRAM controller/Quiddikey	Includes PUFs that are virtually impossible to duplicate or clone and consequently helps in secure key generation and storage and device authentication.
PUF subsystem	Stores the applications keys, which are provisioned to ELS key store, securely.
Cyclic Redundancy Check (CRC)	Provides error detection for all single and double errors, and many multi-bit errors.
PRINCE	Ensures confidentiality protection of the content stored on internal flash. Also ensures both confidentiality and integrity protection of the content stored on external flash. Run-time code integrity protection is ensured during execution from external flash.
Digital Tamper (TDET)	Supports tamper detection.
Code Watchdog Timer (CDOG)	Helps protect the integrity of software by detecting unexpected changes (faults) in the code execution flow.
Digital and analog Glitch Detect (GDET)	A fully digital detector connected to the supply lines to provide glitch detection.
OTPC w 4kb Fuse	An on-chip EFUSE OTP controller from TSMC which supports loading and housing of EFUSE content into shadow registers.
Intrusion and Tamper Response Controller (ITRC)	Provides mechanism to configure the response action for an intrusion event detected by an on-chip security sensor.

Table continues on the next page...

Table 9. Security modules (continued)

Module	Description
Secure AHB bus and AHB Controller (AHBSC)	The device implements a second layer of protection with secure AHB Bus to support secure trusted execution at the system level. The secure AHB Controller provides access policies for all the bus slaves via checker functions.
Memory Block Checker (MBC)	Provides read, write, and execute access control per block to internal flash memory.
TrustZone - M (TZM)	Is an Arm security feature related to hardware-enforced access control mechanisms.

2.5.7 Timer

The following timer modules are available on this chip.

Table 10. Timer modules

Module	Description
Standard Counter/Timer (CTIMER)	Each Counter/timer is designed to count cycles of the CTIMER function clock.
SCTIMER	A powerful and flexible timer module capable of creating complex PWM waveforms with minimal or no CPU intervention.
Multi Rate Timer (MRT)	Provides a repetitive 24-bit interrupt timer with four channels.
Windowed Watchdog Timer (WWDT)	Helps reset or interrupt an erroneous microcontroller within a programmable time.
Micro-Tick (UTICK) Timer	A 31-bit timer that provides a fixed time interval between interrupts.
OS Event Timer (OSTIMER)	A 42-bit Gray code counter.
Enhanced Flex Pulse Width Modulator (PWM)	Generates various switching patterns, including highly sophisticated waveforms, and controls different Switched Mode Power Supplies (SMPS) topologies.
Quadrature Decoder (QDC)	Interfaces to position and speed sensors that are used in industrial motor control applications.
Real Time Clock Subsystem (RTC_SUBSYSTEM)	Supports subsecond and wake timer features.
Real Time Clock (RTC)	Provides time keeping and calendaring functions and additionally provides protection against spurious memory and register updates and battery operation.
Frequency Measurement (FREQME)	Provides high-accuracy frequency measurement function for on-chip and off-chip clocks.
Low-power Timer (LPTMR)	A 16-bit timer or pulse counter with compare feature.

Table continues on the next page...

Table 10. Timer modules (continued)

Module	Description
External Watchdog Monitor (EWM)	Monitors external circuits, as well as the MCU software flow. Also provides a backup mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

2.5.8 Communication

The following communication modules are available on this chip.

Table 11. Communication modules

Module	Description
Universal Serial Bus Full Speed Host and Device Controller (USBFS)	Supports OTG protocol. The OTG implementation in this module provides a device solution for a full speed compliant peripheral along with limited host functionality for implementing a USB 2.0 full speed/low speed compliant embedded host.
Universal Serial Bus 2.0 High-Speed Integrated Physical Layer Interface (USBHS_PHY)	This chip contains USB 2.0 PHY macrocells capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, full-speed (FS) rate of 12 Mbits/s, or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s. The USB PHY supports USB Battery Charger Specification Revision 1.2.
Universal Serial Bus - Device Charger Detect (USBDCD)	Monitors the USB data lines to detect attachment to a USB charging port meeting the USB Battery Charging Specification Rev1.2.
Universal Serial Bus - High Speed Controller (USBHS)	Provides high performance USB functionality that conforms to the Universal Serial Bus Specification, Rev. 2.0.
Ultra Secured Digital Host Controller (uSDHC)	Provides the interface between the host system and the SD/SDIO/MMC cards.
Low-power Flexible Communications Interface (LP_FLEXCOMM)	Provides an option to choose one peripheral function from a choice of several peripherals, such as Low Power Inter-Integrated Circuit (LPI2C), Low Power Serial Peripheral Interface (LPSPI), and Low Power Universal Asynchronous Receive/Transmit (LPUART).
Serial Audio Interface (SAI)	A full-duplex, serial port that allows the chip to communicate with serial devices that implement the Inter-IC sound bus (I2S) and the Intel® AC97 standards.
FlexCAN with FD	A communication controller implementing the Controller Area Network (CAN) protocol according to the ISO 11898-1 standard and CAN 2.0 B protocol specifications.
Smart Card Interface Module (EMVSIM)	Facilitates communication to Smart Cards compatible with the EMV v4.3 standard (Book 1) and Smart Cards compatible with ISO/IEC 7816-3 Standard.

Table continues on the next page...

Table 11. Communication modules (continued)

Module	Description
Ethernet QoS	Enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard.
Flexible Input/Output (FlexIO)	Emulation of a variety of serial/parallel communication protocols. Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions.
Improved Inter-Integrated Circuit (I3C)	An extension of the I2C bus protocol supporting higher speeds.

2.5.9 Human Machine Interface (HMI)

The following HMI modules are available on this chip.

Table 12. HMI modules

Module	Description
General Purpose Input/Output (GPIO)	All GPIO pins support interrupt and DMA request generation.
Port Control (PORT)	Provides support for pad control functions.
Pin Interrupt and Pattern Match (PINT)	Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Uses standard GPIO functions as inputs.
PDM Microphone Interface (MICFIL)	Supports audio delivery from microphones to the processor in several applications, such as mobile telephones.
Touch Sense Interface (TSI)	Provides touch sensing detection on capacitive touch sensors.

2.5.10 Analog

The following analog modules are available on this chip.

Table 13. Analog modules

Module	Description
Analog-to-Digital Converter (ADC)	A dual 16-bit successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.
14-bit Digital-to-Analog Converter (DAC)	A high performance, 14-bit general-purpose DAC the output of which can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.
12-bit Digital-to-Analog Converter	A low power, 12-bit general-purpose DAC the output of which can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.
Low Power Comparator (LPCMP)	Provides a circuit for comparing two analog input voltages.

Table continues on the next page...

Table 13. Analog modules (continued)

Module	Description
Voltage Reference (VREF)	Supplies an accurate voltage output that can be used by internal or external peripherals.
Operational Amplifier (OPAMP)	Supports more than 100 dB amplifier gain.
SINC Filter	Converts external ADC sigma-delta modulator bit stream to data stream.

Chapter 3
Core Overview

3.1 Introduction

This section covers the core modules included in this chip.

NOTE

This chapter is also repeated in the MCX N Security Reference Manual, with differences only in the Peripheral Bridge memory map tables in Peripheral Bridge (PBRG). The sections in the Reference Manual does not show the security relevant information in it.

3.2 CPU0 and CPU1 Cortex-M33 Code and System buses

CPU0 is the primary Cortex-M33 (ver r0p4-00rel0) processor, which supports TrustZone-M, Floating Point Unit (FPU), and Memory Protection Unit (MPU).

The MCX N device also includes a second instance of Cortex-M33, CPU1, which is the secondary Mirco-CM33 intended to offload the main processor to support special dedicated applications. The configuration of this instance does not include MPU, FPU, DSP, ETM, Trustzone-M, Secure Attribution Unit (SAU) or co-processor interface. SYSTICK is supported on both cores.

Cortex-M33 implements a modified Harvard memory architecture using two 32-bit bus interfaces: the Code and System buses. The bus interfaces are activated by address range and can include both instruction fetches and operand data references on a given bus port. (A traditional Harvard architecture strictly separates instruction fetches and operand data references onto specific bus ports regardless of access address.) The Code bus is typically used for instruction fetching and data accesses of PC-relative data, while the system bus is typically used for operand data references to the on-chip and off-chip memories and peripheral accesses. The bus structure fully supports concurrent instruction fetch and data accesses, but the Cortex-M33 implementations can generate both types of references on each bus.

NOTE

It is recommended that performance critical code be located such that it fetches from the Code bus interface as defined by addresses < 0x2000_0000.

3.2.1 Code Bus access

CPU0 and CPU1 code bus can access memory as shown in Bus matrix block diagram . CPU0 code bus access is routed to Low Power Cache (LPCAC) controller. This controller then processes the cacheable accesses as needed, while bypassing the non-cacheable accesses or forwarding the cache write-through and cache miss accesses to the downstream memories through the master port of this cache controller.

3.2.2 System bus access

All System bus accesses are routed to the target address in destination memories through multilayer AHB matrix slave port. See Bus matrix block diagram .

3.2.3 Access control

CPU0 Code and System Bus accesses are checked by the core access control logic, that is, IDAU/SAU and MPU. All requests that miss or bypass the cache are checked by downstream secure AHB bus logic. The caches include protection control signals (HPROT[3:0], which is defined in Arm AHB protocol) and processing domain bits as part of the tags. If a fetch address hits the cache but the protection control and/or domain bits are different, the cache controller forces a miss with the allocate location the same as the address hit location in the cache. This policy allows all the downstream checks to take place, and this new miss is loaded in the cache with the updated protection control and domain bits overwriting the line with the same address. This keeps the cache coherent while always checking accesses that need to see the downstream checks.

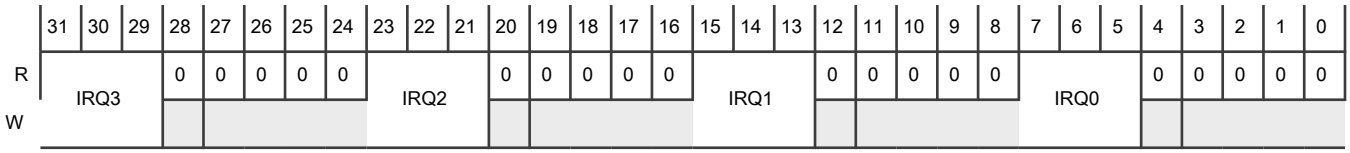
CPU1 access is controlled by secure AHB control.

Read, write, and execute access control to on-chip flash is controlled by Memory Block Checker (MBC). See MBC chapter in MCX N Security Reference Manual.

3.3 Nested Vectored Interrupt Controller (NVIC)

3.3.1 Interrupt priority levels

This device supports 8 priority levels for interrupts. Therefore, in the NVIC each source in the IPR registers contains 3 bits. For example, IPR0 is shown below:



3.3.2 Non-Maskable Interrupt (NMI) configuration

The Non-Maskable Interrupt (NMI) enable bit and source selection bits are implemented for each core in SYSCON NMI Source Select register.

3.3.3 Interrupt channel assignments

The interrupt source assignments are defined in the following table.

- Vector number - the value stored on the stack when an interrupt is serviced.
- IRQ number - non-core interrupt source count, which is the vector number minus 16.

The IRQ number is used within Arm's NVIC documentation.

Table 15. Interrupt Vector Assignments

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0000	0	-	-	-	Cortex-M33	Initial Stack Pointer
0000_0004	1	-	-	-	Cortex-M33	Initial Program Counter
0000_0008	2	-	-	-	Cortex-M33	Non-Maskable Interrupt (NMI)
0000_000C	3	-	-	-	Cortex-M33	Hard Fault
0000_0010	4	-	-	-	Cortex-M33	MemManage Fault
0000_0014	5	-	-	-	Cortex-M33	Bus Fault
0000_0018	6	-	-	-	Cortex-M33	Usage Fault
0000_001C	7	-	-	-	Cortex-M33	Secure fault
0000_0020	8	-	-	-	Reserved	
0000_0024	9	-	-	-	Reserved	
0000_0028	10	-	-	-	Reserved	

Table continues on the next page...

Table 15. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_002C	11	-	-	-	Cortex-M33	Supervisor Call (SVCall)
0000_0030	12	-	-	-	Cortex-M33	Debug Monitor
0000_0034	13	-	-	-	Reserved	
0000_0038	14	-	-	-	Cortex-M33	Pendable request for system service (Pendable SrvReq)
0000_003C	15	-	-	-	Cortex-M33	System Tick Timer
0000_0040	16	0	0	0		ORs IRQs 1 - 155 (OR IRQ[155:1])
0000_0044	17	1	0	0	eDMA_0	eDMA_0_CH0 error or transfer complete
0000_0048	18	2	0	0	eDMA_0	eDMA_0_CH1 error or transfer complete
0000_004C	19	3	0	0	eDMA_0	eDMA_0_CH2 error or transfer complete
0000_0050	20	4	0	1	eDMA_0	eDMA_0_CH3 error or transfer complete
0000_0054	21	5	0	1	eDMA_0	eDMA_0_CH4 error or transfer complete
0000_0058	22	6	0	1	eDMA_0	eDMA_0_CH5 error or transfer complete
0000_005C	23	7	0	1	eDMA_0	eDMA_0_CH6 error or transfer complete
0000_0060	24	8	0	2	eDMA_0	eDMA_0_CH7 error or transfer complete
0000_0064	25	9	0	2	eDMA_0	eDMA_0_CH8 error or transfer complete
0000_0068	26	10	0	2	eDMA_0	eDMA_0_CH9 error or transfer complete
0000_006C	27	11	0	2	eDMA_0	eDMA_0_CH10 error or transfer complete
0000_0070	28	12	0	3	eDMA_0	eDMA_0_CH11 error or transfer complete
0000_0074	29	13	0	3	eDMA_0	eDMA_0_CH12 error or transfer complete
0000_0078	30	14	0	3	eDMA_0	eDMA_0_CH13 error or transfer complete
0000_007C	31	15	0	3	eDMA_0	eDMA_0_CH14 error or transfer complete
0000_0080	32	16	0	4	eDMA_0	eDMA_0_CH15 error or transfer complete
0000_0084	33	17	0	4	GPIO0	GPIO0 interrupt 0
0000_0088	34	18	0	4	GPIO0	GPIO0 interrupt 1
0000_008C	35	19	0	4	GPIO1	GPIO1 interrupt 0
0000_0090	36	20	0	5	GPIO1	GPIO1 interrupt 1
0000_0094	37	21	0	5	GPIO2	GPIO2 interrupt 0
0000_0098	38	22	0	5	GPIO2	GPIO2 interrupt 1
0000_009C	39	23	0	5	GPIO3	GPIO3 interrupt 0
0000_00A0	40	24	0	6	GPIO3	GPIO3 interrupt 1

Table continues on the next page...

Table 15. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_00A4	41	25	0	6	GPIO4	GPIO4 interrupt 0
0000_00A8	42	26	0	6	GPIO4	GPIO4 interrupt 1
0000_00AC	43	27	0	6	GPIO5	GPIO5 interrupt 0
0000_00B0	44	28	0	7	GPIO5	GPIO5 interrupt 1
0000_00B4	45	29	0	7	UTICK0	Micro-Tick Timer interrupt
0000_00B8	46	30	0	7	MRT0	Multi-Rate Timer interrupt
0000_00BC	47	31	0	7	CTIMER0	Standard counter/timer 0 interrupt
0000_00C0	48	32	1	8	CTIMER1	Standard counter/timer 1 interrupt
0000_00C4	49	33	1	8	SCT0	SCTIMER/PWM interrupt
0000_00C8	50	34	1	8	CTIMER2	Standard counter/timer 2 interrupt
0000_00CC	51	35	1	8	LP_FLEXCOMM0	LP_FLEXCOMM0 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00D0	52	36	1	9	LP_FLEXCOMM1	LP_FLEXCOMM1 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00D4	53	37	1	9	LP_FLEXCOMM2	LP_FLEXCOMM2 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00D8	54	38	1	9	LP_FLEXCOMM3	LP_FLEXCOMM3 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00DC	55	39	1	9	LP_FLEXCOMM4	LP_FLEXCOMM4 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00E0	56	40	1	10	LP_FLEXCOMM5	LP_FLEXCOMM5 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00E4	57	41	1	10	LP_FLEXCOMM6	LP_FLEXCOMM6 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00E8	58	42	1	10	LP_FLEXCOMM7	LP_FLEXCOMM7 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00EC	59	43	1	10	LP_FLEXCOMM8	LP_FLEXCOMM8 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00F0	60	44	1	11	LP_FLEXCOMM9	LP_FLEXCOMM8 (LPSPI interrupt or LPI2C interrupt or LPUART Receive/Transmit interrupt)
0000_00F4	61	45	1	11	ADC0	Analog-to-Digital Converter 0 - General Purpose interrupt
0000_00F8	62	46	1	11	ADC1	Analog-to-Digital Converter 1 - General Purpose interrupt
0000_00FC	63	47	1	11	PINT0	Pin Interrupt Pattern Match Interrupt

Table continues on the next page...

Table 15. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0100	64	48	1	12	MICFIL0	Microphone Interface interrupt
0000_0104	65	49	1	12	Reserved	Reserved
0000_0108	66	50	1	12	USBFS0	Universal Serial Bus - Full Speed interrupt
0000_010C	67	51	1	12	USBDCD0	Universal Serial Bus - Device Charge Detect interrupt
0000_0110	68	52	1	13	RTC0	RTC Subsystem interrupt (RTC interrupt or Wake timer interrupt)
0000_0114	69	53	1	13	SmartDMA	SmartDMA_IRQ
0000_0118	70	54	1	13	MAILBOX0	Inter-CPU Mailbox interrupt0 for CPU0 Inter-CPU Mailbox interrupt1 for CPU1
0000_011C	71	55	1	13	CTIMER3	Standard counter/timer 3 interrupt
0000_0120	72	56	1	14	CTIMER4	Standard counter/timer 4 interrupt
0000_0124	73	57	1	14	OSTIMER0	OS event timer interrupt
0000_0128	74	58	1	14	FlexSPI0	Flexible Serial Peripheral Interface interrupt
0000_012C	75	59	1	14	SAI0	Serial Audio Interface 0 interrupt
0000_0130	76	60	1	15	SAI1	Serial Audio Interface 1 interrupt
0000_0134	77	61	1	15	uSDHC0	Ultra Secured Digital Host Controller interrupt
0000_0138	78	62	1	15	CAN0	Controller Area Network 0 interrupt
0000_013C	79	63	1	15	CAN1	Controller Area Network 1 interrupt
0000_0140	80	64	2	16	Reserved	—
0000_0144	81	65	2	16	Reserved	—
0000_0148	82	66	2	16	USBHS1_PHY	USBHS DCD or USBHS Phy interrupt
0000_014C	83	67	2	16	USBHS1	USB High Speed OTG Controller interrupt
0000_0150	84	68	2	17	—	Refer MCX N Security Reference Manual
0000_0154	85	69	2	17	Reserved	—
0000_0158	86	70	2	17	PLU0	Programmable Logic Unit interrupt
0000_015C	87	71	2	17	FREQME0	Frequency Measurement interrupt
0000_0160	88	72	2	18	—	Refer MCX N Security Reference Manual
0000_0164	89	73	2	18	—	Refer MCX N Security Reference Manual
0000_0168	90	74	2	18	—	Refer MCX N Security Reference Manual
0000_016C	91	75	2	18	—	Refer MCX N Security Reference Manual

Table continues on the next page...

Table 15. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0170	92	76	2	19	POWERQUAD0	PowerQuad interrupt
0000_0174	93	77	2	19	eDMA_1	eDMA_1_CH0 error or transfer complete
0000_0178	94	78	2	19	eDMA_1	eDMA_1_CH1 error or transfer complete
0000_017C	95	79	2	19	eDMA_1	eDMA_1_CH2 error or transfer complete
0000_0180	96	80	2	20	eDMA_1	eDMA_1_CH3 error or transfer complete
0000_0184	97	81	2	20	eDMA_1	eDMA_1_CH4 error or transfer complete
0000_0188	98	82	2	20	eDMA_1	eDMA_1_CH5 error or transfer complete
0000_018C	99	83	2	20	eDMA_1	eDMA_1_CH6 error or transfer complete
0000_0190	100	84	2	21	eDMA_1	eDMA_1_CH7 error or transfer complete
0000_0194	101	85	2	21	eDMA_1	eDMA_1_CH8 error or transfer complete
0000_0198	102	86	2	21	eDMA_1	eDMA_1_CH9 error or transfer complete
0000_019C	103	87	2	21	eDMA_1	eDMA_1_CH10 error or transfer complete
0000_01A0	104	88	2	22	eDMA_1	eDMA_1_CH11 error or transfer complete
0000_01A4	105	89	2	22	eDMA_1	eDMA_1_CH12 error or transfer complete
0000_01A8	106	90	2	22	eDMA_1	eDMA_1_CH13 error or transfer complete
0000_01AC	107	91	2	22	eDMA_1	eDMA_1_CH14 error or transfer complete
0000_01B0	108	92	2	23	eDMA_1	eDMA_1_CH15 error or transfer complete
0000_01B4	109	93	2	23	—	Refer MCX N Security Reference Manual
0000_01B8	110	94	2	23	—	Refer MCX N Security Reference Manual
0000_01BC	111	95	2	23	I3C0	Improved Inter Integrated Circuit interrupt 0
0000_01C0	112	96	3	24	I3C1	Improved Inter Integrated Circuit interrupt 1
0000_01C4	113	97	3	24	NPU	NPU interrupt
0000_01C8	114	98	3	24	—	Refer MCX N Security Reference Manual
0000_01CC	115	99	3	24	VBAT0	VBAT interrupt (VBAT interrupt or digital tamper interrupt)
0000_01D0	116	100	3	25	EWM0	External Watchdog Monitor interrupt
0000_01D4	117	101	3	25	TSI0	TSI End of Scan interrupt
0000_01D8	118	102	3	25	TSI0	TSI End of Scan interrupt
0000_01DC	119	103	3	25	EMVSIM0	EMVSIM0 interrupt
0000_01E0	120	104	3	26	EMVSIM1	EMVSIM1 interrupt

Table continues on the next page...

Table 15. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_01E4	121	105	3	26	FlexIO0	Flexible Input/Output interrupt
0000_01E8	122	106	3	26	DAC0	12-bit Digital-to-Analog Converter 0 - General Purpose interrupt
0000_01EC	123	107	3	26	DAC1	12-bit Digital-to-Analog Converter 1 - General Purpose interrupt
0000_01F0	124	108	3	27	DAC2	14-bit Digital-to-Analog Converter interrupt
0000_01F4	125	109	3	27	CMP0	Comparator0 interrupt
0000_01F8	126	110	3	27	CMP1	Comparator1 interrupt
0000_01FC	127	111	3	27	CMP2	Comparator2 interrupt
0000_0200	128	112	3	28	PWM0	FlexPWM0_reload_error interrupt
0000_0204	129	113	3	28	PWM0	FlexPWM0_fault interrupt
0000_0208	130	114	3	28	PWM0	FlexPWM0 Submodule 0 capture/compare/ reload interrupt
0000_020C	131	115	3	28	PWM0	FlexPWM0 Submodule 1 capture/compare/ reload interrupt
0000_0210	132	116	3	29	PWM0	FlexPWM0 Submodule 2 capture/compare/ reload interrupt
0000_0214	133	117	3	29	PWM0	FlexPWM0 Submodule 3 capture/compare/ reload interrupt
0000_0218	134	118	3	29	PWM1	FlexPWM1_reload_error interrupt
0000_021C	135	119	3	29	PWM1	FlexPWM1_fault interrupt
0000_0220	136	120	3	30	PWM1	FlexPWM1 Submodule 0 capture/compare/ reload interrupt
0000_0224	137	121	3	30	PWM1	FlexPWM1 Submodule 1 capture/compare/ reload interrupt
0000_0228	138	122	3	30	PWM1	FlexPWM1 Submodule 2 capture/compare/ reload interrupt
0000_022C	139	123	3	30	PWM1	FlexPWM1 Submodule 3 capture/compare/ reload interrupt
0000_0230	140	124	3	31	QDC0	QDC0_Compare interrupt
0000_0234	141	125	3	31	QDC0	QDC0_Home interrupt
0000_0238	142	126	3	31	QDC0	QDC0_WDG_IRQ/SAB interrupt
0000_023C	143	127	3	31	QDC0	QDC0_IDX interrupt
0000_0240	144	128	4	32	QDC1	QDC1_Compare interrupt

Table continues on the next page...

Table 15. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0244	145	129	4	32	QDC1	QDC1_Home interrupt
0000_0248	146	130	4	32	QDC1	QDC1_WDG_IRQ/SAB interrupt
0000_024C	147	131	4	32	QDC1	QDC1_IDX interrupt
0000_0250	148	132	4	33	—	Refer MCX N Security Reference Manual
0000_0254	149	133	4	33	BSP32	CoolFlux BSP32 interrupt
0000_0258	150	134	4	33	—	Refer MCX N Security Reference Manual
0000_025C	151	135	4	33	—	Refer MCX N Security Reference Manual
0000_0260	152	136	4	34	ERM0	ERM Single Bit error interrupt
0000_0264	153	137	4	34	ERM0	ERM Multi Bit error interrupt
0000_0268	154	138	4	34	FMU0	Flash Management Unit interrupt
0000_026C	155	139	4	34	ENET0	Ethernet QoS interrupt
0000_0270	156	140	4	35	ENET0	Ethernet QoS power management interrupt
0000_0274	157	141	4	35	ENET0	Ethernet QoS MAC LPI interrupt
0000_0278	158	142	4	35	SINC0	SINC Filter interrupt
0000_027C	159	143	4	35	LPTMR0	Low Power Timer 0 interrupt
0000_0280	160	144	4	36	LPTMR1	Low Power Timer 1 interrupt
0000_0284	161	145	4	36	SCG0	System Clock Generator interrupt
0000_0288	162	146	4	36	SPC0	System Power Controller interrupt
0000_028C	163	147	4	36	WUU0	Wake Up Unit interrupt
0000_0290	164	148	4	37	PORT	PORT0~5 EFT interrupt
0000_0294	165	149	4	37	ETB0	ETB counter expires interrupt
0000_0298	166	150	4	37	Reserved	—
0000_029C	167	151	4	37	Reserved	—
0000_02A0	168	152	4	38	WWDT0	Windowed Watchdog Timer 0 interrupt
0000_02A4	169	153	4	38	WWDT1	Windowed Watchdog Timer 1 interrupt
0000_02A8	170	154	4	38	CMC0	Core Mode Controller interrupt
0000_02AC	171	155	4	38	CTI0	Cross Trigger Interface (CTI) interrupt **Note:** CTI interrupt is generated by its own CPU. CPU0 at slot 155 is CPU0 CTI interrupt. CPU1 at slot 155 is CPU1 CTI interrupt.

3.4 Implementation Defined Attribution Unit (IDAU)

MCX N implements a simple attribution unit that divides whole memory map into secure or non-secure regions. All peripherals and memories are aliased at two locations.

- Address 0x0000_0000 to 0x1FFF_FFFF
  - Non-Secure always
- Address 0x2000_0000 to 0xFFFF_FFFF
  - If Address Bit_28 = 0 Non-Secure
  - If Address Bit_28 = 1 Secure

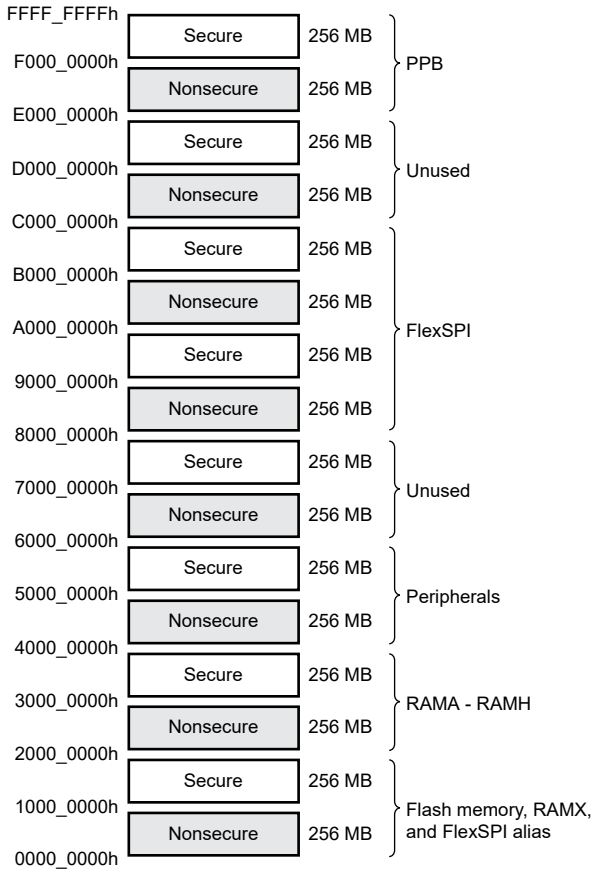


Figure 4. IDAU implementation

Chapter 4
Memory

4.1 Memory architecture

The memory architecture supports a wide range of target applications and maximizes performance while maintaining low power consumption. The memory system of the device includes SRAM, ROM, internal flash, and external memory.

4.2 System memory map

The following table shows the chip's high-level memory map.

Additionally, the various sections are split into "Secure" and "Non-Secure" apertures for each IP. Although these apertures address the same IP, access through the "Secure" aperture can only be performed by code with the appropriate security settings.

NOTE

The secure space is the alias of the non-secure region.

Table 16. System memory map

Start address	End address	TZ-M State[1]	Description	Size	Cached
0000_0000	001F_FFFF	Non-Secure	Program flash **Note:** Program flash has two flash memory arrays/ banks of up to 1 MB each.	2 MB	LPCAC
0020_0000	00FF_FFFF		Reserved	—	—
0100_0000	0100_7FFF		Flash Bank 0 IFR 0	32 KB	No
0100_8000	0100_FFFF		Flash Bank 1 IFR 0	32 KB	No
0101_0000	010F_FFFF		Reserved	—	—
0110_0000	0110_1FFF		Flash Bank 0 IFR 1	8 KB	No
0110_2000	0110_3FFF		Flash Bank 1 IFR 1	8 KB	No
0110_4000	02FF_FFFF		Reserved	—	—
0300_0000	0303_FFFF		ROM-BOOT	256 KB	No
0304_0000	03FF_FFFF		Reserved	—	—
0400_0000	0401_7FFF		RAMX	96 KB	No
0401_8000	07FF_FFFF		Reserved	—	—
0800_0000	0FFF_FFFF		Flexible Serial Peripheral Interface (FlexSPI) **Note:** The 128 MB FlexSPI at address 0x0800_0000 is aliased to the system memory map FlexSPI region at address 0x8000_0000 - 0x87FF_FFFF.	128 MB	LPCAC/ CACHE64
1000_0000	101F_FFFF	Secure	Program Flash	2 MB	LPCAC

Table continues on the next page...

Table 16. System memory map (continued)

Start address	End address	TZ-M State[1]	Description	Size	Cached
			Note: Program flash has two flash memory arrays/ banks of up to 1 MB each.		
1020_0000	10FF_FFFF		Reserved	—	—
1100_0000	1100_7FFF		Flash Bank 0 IFR 0	32 KB	No
1100_8000	1100_FFFF		Flash Bank 1 IFR 0	32 KB	No
1101_0000	110F_FFFF		Reserved	—	—
1110_0000	1110_1FFF		Flash Bank 0 IFR 1	8 KB	No
1110_2000	1110_3FFF		Flash Bank 1 IFR 1	8 KB	No
1110_4000	12FF_FFFF		Reserved	—	—
1300_0000	1303_FFFF		ROM-BOOT	256 KB	No
1304_0000	13FF_FFFF		Reserved	—	—
1400_0000	1401_7FFF		RAMX	96 KB	No
1401_8000	17FF_FFFF		Reserved	—	—
1800_0000	1FFF_FFFF		Flexible Serial Peripheral Interface (FlexSPI) **Note:** The 128 MB FlexSPI at address 0x1800_0000 is aliased to the system memory map FlexSPI region at address 0x9000_0000-0x97FF_FFFF.	128 MB	LPCAC/ CACHE64
2000_0000	2000_7FFF	Non-Secure	RAMA	32 KB	No
2000_8000	2000_FFFF		RAMB	32 KB	No
2001_0000	2001_FFFF		RAMC	64 KB	No
2002_0000	2002_FFFF		RAMD	64 KB	No
2003_0000	2003_FFFF		RAME	64 KB	No
2004_0000	2004_FFFF		RAMF	64 KB	No
2005_0000	2005_FFFF		RAMG[2]	64 KB	No
2006_0000	2006_7FFF		RAMH[3,2]	32 KB	No
2006_8000	27FF_FFFF		Reserved	—	—
2800_0000	2FFF_FFFF		Reserved	—	—
3000_0000	3000_7FFF	Secure	RAMA	32 KB	No
3000_8000	3000_FFFF		RAMB	32 KB	No
3001_0000	3001_FFFF		RAMC	64 KB	No
3002_0000	3002_FFFF		RAMD	64 KB	No
3003_0000	3003_FFFF		RAME	64 KB	No

Table continues on the next page...

Table 16. System memory map (continued)

Start address	End address	TZ-M State[1]	Description	Size	Cached
3004_0000	3004_FFFF		RAMF	64 KB	No
3005_0000	3005_FFFF		RAMG[2]	64 KB	No
3006_0000	3006_7FFF		RAMH[2,3]	32 KB	No
3006_8000	37FF_FFFF		Reserved	—	—
3800_0000	3FFF_FFFF	Secure	Reserved	—	—
4000_0000	4005_FFFF	Non-Secure	Peripheral Bridge 0 - PBRG0	384 KB	No
4006_0000	4007_FFFF		Reserved	—	—
4008_0000	4009_FFFF	Non-Secure	Peripheral Bridge 1 - PBRG1	128 KB	No
400A_0000	400B_FFFF		Peripheral Bridge 2 - PBRG2	128 KB	No
400C_0000	400D_FFFF		Peripheral Bridge 3 - PBRG3	128 KB	No
400E_0000	400F_FFFF		Reserved	—	—
4010_0000	4013_FFFF	Non-Secure	Peripheral Bridge 4 - PBRG4	256 KB	No
4014_0000	4FFF_FFFF		Reserved	—	—
5000_0000	5005_FFFF	Secure	Peripheral Bridge 0 - PBRG0	384 KB	No
5006_0000	5007_FFFF		Reserved	—	—
5008_0000	5009_FFFF	Secure	Peripheral Bridge 1 - PBRG1	128 KB	No
500A_0000	500B_FFFF		Peripheral Bridge 2 - PBRG2	128 KB	No
500C_0000	500D_FFFF		Peripheral Bridge 3 - PBRG3	128 KB	No
500E_0000	500F_FFFF		Reserved	—	—
5010_0000	5013_FFFF	Secure	Peripheral Bridge 4 - PBRG4	256 KB	No
5014_0000	5FFF_FFFF		Reserved	—	—
6000_0000	6FFF_FFFF		Reserved	—	—
7000_0000	7FFF_FFFF		Reserved	—	—
8000_0000	8FFF_FFFF	Non-Secure	Flexible Serial Peripheral Interface (FlexSPI)	256 MB	CACHE64
9000_0000	9FFF_FFFF	Secure	Flexible Serial Peripheral Interface (FlexSPI)	256 MB	CACHE64
A000_0000	AFFF_FFFF	Non-Secure	Flexible Serial Peripheral Interface (FlexSPI)	256 MB	CACHE64
B000_0000	BFFF_FFFF	Secure	Flexible Serial Peripheral Interface (FlexSPI)	256 MB	CACHE64
C000_0000	CFFF_FFFF		Reserved	—	—
D000_0000	DFFF_FFFF		Reserved	—	—
E000_0000	E003_FFFF		Private peripheral bus (internal)	256 KB	No
E004_0000	E00F_FFFF		Private peripheral bus (external) (includes NVIC and SYSTICK timer)	768 KB	No

Table continues on the next page...

Table 16. System memory map (continued)

Start address	End address	TZ-M State[1]	Description	Size	Cached
E010_0000	FFFF_FFFF		Reserved	—	—

1. This is the default Targeted Security Attribute for the memory region at reset. It is set by the IDAU and the SAU configuration. The attribute listed is the intended use case for users of TrustZone-M (TZM) security function.
2. On device part numbers that support optional ECC, RAMG and RAMH can be used to provide ECC data for other RAM blocks. RAMG and RAMH are not accessible as regular memory when used for ECC, so their availability in the memory map depends on the selected ECC configuration.
3. On device part numbers that support optional ECC, RAMB and RAMX ECC is enabled by default. This means that RAMH is used to provide ECC and is not accessible as regular RAM by default.

NOTE

For detailed connection between masters and slaves, see Bus matrix block diagram.

4.3 Cache

To help increase system performance, there are several caches included in the device.

The 16 KB Low Power Cache Controller (LPCAC) is connected to the Code bus of the primary CM33 core. Contents of this cache are only visible to the CM33 core. The LPCAC can be used to cache CM33 access to the program flash (0x0000_0000-0x001F_FFFF and 0x1000_0000-0x101F_FFFF) and FlexSPI (0x0800_0000-0x0FFF_FFFF and 0x1800_0000-0x1FFF_FFFF) code bus memory regions. The LPCAC chapter provides the functional description of the cache, but the SYSCON LPCAC Control (LPCAC_CTRL) register is used to control operation of the cache.

The Flash controller includes a small (64 byte) cache. Because the cache is a part of the memory controller, an access from any master to the flash can potentially be cached. The SYSCON NVM Control (NVM_CTRL) register is used to control and enable the FMC cache.

The 16 KB cache controller (CACHE64) is interfaced directly to the FlexSPI memory controller. Because the CACHE64 is integrated at the controller, an access from any master to the FlexSPI can be cached as long as the corresponding address is configured as cacheable. The CACHE64_POLSEL module is used to define the cache policy (non-cacheable, write-thru cacheable, or write-back cacheable) for up to three regions in the FlexSPI. The CACHE64_POLSEL and CACHE64 use a remapping scheme to map the FlexSPI system memory map regions (0x0800_0000-0x0FFF_FFFF, 0x1800_0000-0x1FFF_FFFF, 0x8000_0000-0x8FFF_FFFF, 0x9000_0000-0x9FFF_FFFF, 0xA000_0000-0xAFFF_FFFF, and 0xB000_0000-0xBFFF_FFFF) to a single, contiguous 512 Mbyte region used to access the FlexSPI. Refer to Table 35 for details on the remapping.

4.4 SRAM

Devices in this family support up to 512 KB of on-chip RAM (with ECC disabled). Refer to the device data sheet to determine the RAM size for a given part number. Refer to Table 17 for the RAM configurations.

RAMX (up to 96 KB) is connected to the CM33 Code buses. RAMX is the preferred RAM block to use for code storage.

RAMA which is always four 8 KB banks (32 KB total) is the preferred RAM block to use for data retention. The RAMA banks can be retained in device low-power modes. It can also optionally be powered from VBAT using LDO_RAM. To optimize power consumption, RAMA is split into 4 banks, where the low-power mode and VBAT retention for each bank is individually programmable. Low power configuration (LDO_RAM enable/disable and bank retention) are controlled by the VBAT module.

The other RAM blocks and partitions (other than RAMA) all have independent power switches that can be turned on/off depending on the application's RAM needs. CMC_SRAMDIS0 can be used to completely power gate a RAM partition (applies for all power modes). CMC_SRAMRET0 can be used to turn off the periphery of RAM partitions while retaining the contents of those RAMs during Power Down mode.

4.4.1 RAM ECC

RAMA always supports software configurable ECC (enabled by default). Each 8 KB RAM bank support 32+7 ECC, which provides one bit correction and two bits detection capability.

Depending on the specific device part number, ECC might be supported for the other RAM blocks. Refer to the device datasheet to determine if a given part number supports ECC for RAMs other than RAMA. For non-RAMA blocks, the ECC bits are provided by re-purposing RAMG and/or RAMH sub-blocks to provide ECC. RAMG and RAMH are used for ECC bits even for devices that don't support the full 512 KB of RAM, where RAMG and RAMH are not available as normal RAM memory. For devices with less than 512 KB of total RAM, other RAM blocks are disabled as ECC is enabled to keep the total amount of RAM being used including ECC blocks balanced. Therefore, enabling optional ECC will decrease the overall available memory as RAMG, RAMH, or other memory blocks become inaccessible.

NOTE

The SYSCON_ECC_ENABLE_CTRL bits must be set sequentially starting with bit 0 and working up to bit 3. Enabling ECC for RAMA and RAMB/RAMX is allowed. It is not legal to enable ECC for RAMA and RAMD/RAMC and leave ECC disabled on RAMB/RAMX (setting bit 0 and bit 2 but skipping bit 1).

The SYSCON's ECC_ENABLE_CTRL register is used to configure ECC functionality. ECC is always enabled by default for RAMA. RAMX/RAMB ECC is enabled, by default, for part numbers that support ECC. Other RAM blocks default to ECC disabled, even on ECC enabled part numbers.

Refer to the table below for details on the RAM configurations.

Table 17. RAM configurations

RAM Block	Start Address	End Address	Sub-Block	Size (KB)	Memory block availability based on device total memory size		ECC RAM	SYSCON ECC_ENABLE_CTRL	CMC SRAMDIS0/ SRAMRET0[1]
					352 KB (no ECC)/288 KB (with ECC)	512 KB (no ECC)/416 KB (with ECC)			
RAMX	0400_0000	0400_7FFF	RAMX0	32	Y	Y	RAMH1	bit 1	bit 0
	0400_8000	0400_FFFF	RAMX1	32	Y	Y			bit 1
	0401_0000	0401_7FFF	RAMX2	32	Y	Y			bit 2
RAMA	2000_0000	2000_1FFF	RAMA0	8	Y	Y	n/a - ECC always available	bit 0	VBAT LDORAMC[8]
	2000_2000	2000_3FFF	RAMA1	8	Y	Y			VBAT LDORAMC[9]
	2000_4000	2000_5FFF	RAMA2	8	Y	Y			VBAT LDORAMC[10]
	2000_6000	2000_7FFF	RAMA3	8	Y	Y			VBAT LDORAMC[11]
RAMB	2000_8000	2000_FFFF	RAMB0	32	Y	Y	RAMH0	bit 1	bit 3
RAMC	2001_0000	2001_7FFF	RAMC0	32	Y	Y	RAMG3	bit 2	bit 4

Table continues on the next page...

Table 17. RAM configurations (continued)

RAM Block	Start Address	End Address	Sub-Block	Size (KB)	Memory block availability based on device total memory size		ECC RAM	SYSCON ECC_ENABLE_CTRL	CMC SRAMDIS0/ SRAMRET0[1]
					352 KB (no ECC)/288 KB (with ECC)	512 KB (no ECC)/416 KB (with ECC)			
	2001_8000	2001_FFFF	RAMC1	32	Y	Y			bit 5
RAMD	2002_0000	2002_7FFF	RAMD0	32	Y	Y	RAMG2		bit 6
	2002_8000	2002_FFFF	RAMD1	32	Y	Y			bit 7
RAME	2003_0000	2003_7FFF	RAME0	32	Y[2]	Y	RAMG1	bit 3	bit 8
	2003_8000	2003_FFFF	RAME1	32	Y[3]	Y			bit 9
RAMF	2004_0000	2004_7FFF	RAMF0	32	N	Y	RAMG0		bit 10
	2004_8000	2004_FFFF	RAMF1	32	N	Y			bit 11
RAMG	2005_0000	2005_3FFF	RAMG0	16	N	Y[4]	-	-	bit 12
	2005_4000	2005_7FFF	RAMG1	16	N	Y[4]			
	2005_8000	2005_BFFF	RAMG2	16	ECC[5]	Y[4]			bit 13
	2005_C000	2005_FFFF	RAMG3	16	ECC[5]	Y[4]			
RAMH	2006_0000	2006_1FFF	RAMH0	8	ECC[5]	Y			bit 14
	2006_2000	2006_7FFF	RAMH1	24	ECC[5]	Y[4]			

1. See SRAMDIS0 register and SRAMRET0 register.
2. Block becomes inaccessible when RAMD and RAMC ECC is enabled (SYSCON_ECC_ENABLE_CTRL[RAMD_RAMC_ECC_ENABLE] = 1).
3. Block becomes inaccessible when RAMB and RAMX ECC is enabled (SYSCON_ECC_ENABLE_CTRL[RAMB_RAMX_ECC_ENABLE] = 1).
4. Block is not accessible if ECC is enabled for its pair.
5. Block is not accessible as part of main memory but is used if ECC is enabled for its pair.

NOTE

For parts that support ECC on RAM blocks other than RAMA, ECC is enabled by default for the RAMB and RAMX blocks. This means the RAMH block is not accessible by default.

NOTE

While the RAMA-RAMH blocks are implemented as contiguous regions in the system memory map, each of the RAM blocks uses a different physical AHB slave port for access. This means that misaligned or burst accesses across the boundary from one RAM block to another are not allowed.

4.4.2 RAM ECC Error

The Error Recording Module (ERM) provides information and optional interrupt notification on SRAM error events associated with ECC. The syndrome and error address information is captured along with error event in ERM registers.

The Error Injection Module (EIM) can be used to induce artificial errors in the RAM ECC. EIM can inject single-bit and multi-bit inversions on data. Injecting faults on memory accesses can be used to exercise the SEC-DED ECC function of the related system.

4.4.3 RAM clock gating

To reduce power consumption, the RAM blocks support an auto clock gating feature. When auto clock gating is enabled, if the RAM block is not accessed for 16 bus clocks, then the clock to the RAM block is automatically gated off. If the clock is off, then there is a one bus cycle delay for the next access to the RAM block.

The auto clock gating feature is configurable on a per RAM block basis. The auto clock gating function is configured by the SYSCON's AUTOCLKGATEOVERRIDE and AUTOCLKGATEOVERRIDEC registers. By default, auto clock gating is enabled for the RAMX and RAMA blocks. The auto clock gating feature should be disabled, for RAM blocks that are used for code and data sections that require time-critical or deterministic execution.

In addition to the auto clock gating, the clock for each RAM block other than RAMX and RAMA can be manually disabled if the RAM block is not used or not used in a particular mode. The clock gate controls for the RAM blocks are found in the SYSCON_AHBCLKCTRL0 register. By default, the clocks are disabled for most RAM blocks; however, the ROM will enable the clock for all RAM blocks when booting.

NOTE

If ECC is enabled for blocks other than RAMA, then the clock for the corresponding ECC RAM must be enabled.

4.5 Read Only Memory (ROM)

The internal ROM memory is used to store the boot code and time-critical software library routines. After a reset, the Cortex-M33 processor starts its code execution from this memory.

4.5.1 ROM size

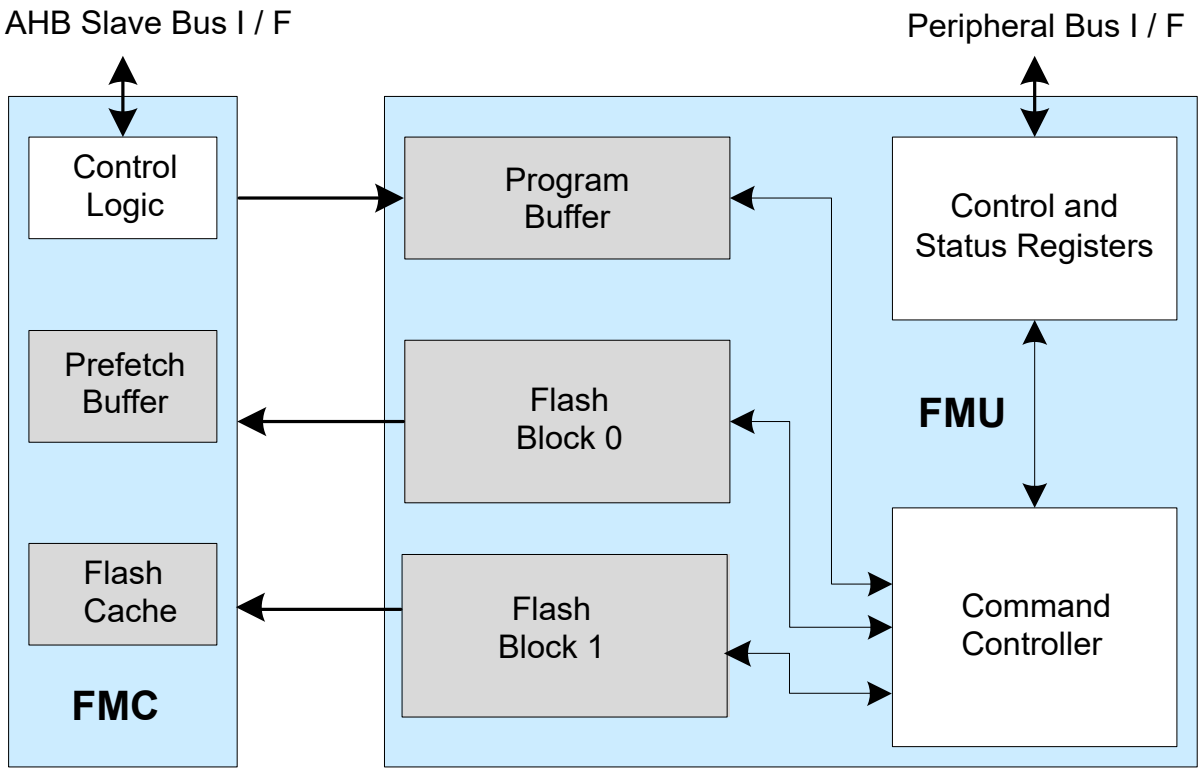
The Boot ROM of this device is 256 KB.

4.6 Internal flash

This device embeds up to 2 MB of flash. It is implemented as 2 x 1MB flash block instances.

Refer to the device datasheet to determine the flash size of a given part number.

The following figure shows the high-level block diagram of the on-chip flash.



4.6.1 Flash memory controller

The Flash Memory Controller (FMC) manages accesses performed by the bus masters of the system to the flash memory. The FMC accelerates flash memory transfers to allow program code execution at higher clock frequency than flash memory.

The FMC provides two separate mechanisms for accelerating read operations to the flash memory:

- A 128-bit prefetch buffer, which can prefetch the next 128-bit flash memory location.
- A 64-Byte cache organized as a one set, four-way associative cache with 128-bit (or 16-Byte) size entries.

Note the FMC and its cache and buffers has no visibility into flash memory erase and program cycles because the Flash Memory module manages them directly.

The speculation logic is tuned to work with the flash cache enabled. If the flash cache is present, the speculation logic assumes it may immediately request the next sequential flash phrase as soon as any access hits the speculation buffer since this will cause the phrase to be loaded in the flash cache. If the flash cache is present and disabled, the speculation logic may move to the next flash phrase before the current flash phrase in the speculation buffer is fully accessed, causing degradation in the speculation logic performance. For best performance on implementation which have both flash cache and speculation logic options available, enable the flash cache whenever the speculation logic is enabled.

4.6.1.1 Prefetch buffer

When speculative reads are enabled, the FMC immediately requests the next sequential address after a read completes. The next 128-bit memory location is read. The speculative prefetch mechanism improves performance by reducing or even eliminating wait states when accessing sequential code and/or data.

The FMC provides invalidation control for the prefetch buffer but the SYSCON's NVM_CTRL register is used to enable and configure speculative prefetching.

While NVM_CTRL[DIS_DATA_SPEC] and NVM_CTRL[DIS_FLASH_SPEC] are cleared by default, the operation of these fields interacts with the NVM_CTRL[DIS_MBECC_ERR_DATA] and NVM_CTRL[DIS_MBECC_ERR_INST].

NVM_CTRL[DIS_MBECC_ERR_DATA] is set by default which disables the flash speculation even though NVM_CTRL[DIS_FLASH_SPEC] is cleared. For best performance, NVM_CTRL[DIS_MBECC_ERR_DATA] should be cleared early on in startup code.

4.6.1.2 Flash cache

Cache memory stores already fetched data. This code is immediately available for repeated execution without any wait states, if needed.

The FMC provides controls for flash replacement algorithm, lock per way, and invalidation per way. The ways are numbered 0-3, and the sets are numbered 0-3. The cache supports Least Recently Used (LRU) replacement algorithm per set across all 4 ways. The SYSCON's NVM_CTRL register is used to enable/disable the cache and set other configurations. The flash cache is enabled by default.

The cache entries, both data and tag/valid, can be read at any time.

Software is required to maintain memory coherence when any segment of the flash cache is programmed. For example, all buffer data associated with the reprogrammed flash should be invalidated. Accordingly, cache program visible writes must occur after a programming or erase event is completed and before the new memory image is accessed.

The access to on-chip flash is checked in the Memory Block Controller (MBC).

4.7 External memory

This device has one FlexSPI subsystem supporting Octal and Quad SPI memory devices. The FlexSPI instance is primarily intended for code execution from off-chip SPI NOR and NAND flash memory. The FlexSPI interfaces also support HyperFlash, HyperRAM and Xccela memory types.

The FlexSPI interface supports execute-in-place and on-the-fly decryption using the IPED with GCM module. The FlexSPI interface includes a 16 KB cache with a CACHE64 AHB-cache controller. The CACHE64_POLSEL module is provided at the CACHE64 interface to enable different caching policies for different address regions. These policies include:

- Write-back
- Write-through
- Non-cached

4.8 Peripheral Bridge (PBRG)

The Peripheral Bridge (PBRG) is the portion of the bus fabric that connects the peripherals to the processor elements. Each peripheral has a base address where the processor elements can access them. The following sections provide the memory map of the peripherals connected to each of the Peripheral Bridges.

4.8.1 Peripheral Bridge 0 (PBRG0) memory map

Table 18. Peripheral bridge 0

Base address	Slot	Module	Alias
4000_0000	**APB 0**		
4000_0000	0	System controller	SYSCON
4000_1000	1	Reserved	—
4000_2000	2	Reserved	—
4000_3000	3	Reserved	—
4000_4000	4	Pin Interrupt and Pattern Match	PINT0

Table continues on the next page...

Table 18. Peripheral bridge 0 (continued)

Base address	Slot	Module	Alias
4000_5000	5	Reserved	—
4000_6000	6	Input multiplexing	INPUTMUX0
4000_7000	7	Reserved	—
4000_8000	8	Reserved	—
4000_9000	9	Reserved	—
4000_A000	10	Reserved	—
4000_B000	11	Reserved	—
4000_C000	12	Standard counter/timers	CTIMER0
4000_D000	13	Standard counter/timers	CTIMER1
4000_E000	14	Standard counter/timers	CTIMER2
4000_F000	15	Standard counter/timers	CTIMER3
4001_0000	16	Standard counter/timers	CTIMER4
4001_1000	17	Frequency Measurement Unit	FREQME0
4001_2000	18	Micro-Tick Timer	UTICK0
4001_3000	19	Multi-Rate Timer	MRT0
4001_4000	20	Reserved	—
4001_5000	21	Reserved	—
4001_6000	22	Windowed Watchdog Timer	WWDT0
4001_7000	23	Windowed Watchdog Timer	WWDT1
4001_8000	24	Reserved	—
4001_9000	25	Reserved	—
4001_A000	26	Reserved	—
4001_B000	27	CACHE64 Policy Select	CACHE64_POLSEL0
4001_C000	28	Reserved	—
4001_D000	29	Reserved	—
4001_E000	30	Reserved	—
4001_F000	31	Reserved	—
4002_0000	APB 1		
4002_0000	0	Reserved	—
4002_1000	1	Improved Inter-Integrated Circuit	I3C0
4002_2000	2	Improved Inter-Integrated Circuit	I3C1

Table continues on the next page...

Table 18. Peripheral bridge 0 (continued)

Base address	Slot	Module	Alias
4002_3000	3	Reserved	—
4002_4000	4	Refer device's Security Reference Manual	—
4002_5000	5	Refer device's Security Reference Manual	—
4002_6000	6	Refer device's Security Reference Manual	—
4002_7000	7	Reserved	—
4002_8000	8	Reserved	—
4002_9000	9	Reserved	—
4002_A000	10	Reserved	—
4002_B000	11	Refer device's Security Reference Manual	—
4002_C000	12	Refer device's Security Reference Manual	—
4002_D000	13	Refer device's Security Reference Manual	—
4002_E000	14	Refer device's Security Reference Manual	—
4002_F000	15	Refer device's Security Reference Manual	—
4003_0000	16	Reserved	—
4003_1000	17	Refer device's Security Reference Manual	—
4003_2000	18	CoolFlux BSP32	BSP32
4003_3000	19	SmartDMA	SmartDMA
4003_4000	20	Programable Logic Unit	PLU0
4003_5000	21	Reserved	—
4003_6000	22	Reserved	—
4003_7000	23	Reserved	
4003_8000	24	Reserved	—
4003_9000	25	Reserved	—
4003_A000	26	Reserved	—
4003_B000	27	Reserved	—
4003_C000	28	Reserved	—
4003_D000	29	Reserved	—
4003_E000	30	Reserved	—
4003_F000	31	Reserved	—
4004_0000	**AIPS 0**		
4004_0000	0	General Purpose Input/Output	GPIO5
4004_1000	1	General Purpose Input/Output (alias 1) - alias of GPIO 5	GPIO5_alias1

Table continues on the next page...

Table 18. Peripheral bridge 0 (continued)

Base address	Slot	Module	Alias
4004_2000	2	Port5	PORT5
4004_3000	3	Flash Management Unit	FMU0
4004_4000	4	System Clock Generator	SCG0
4004_5000	5	System Power Controller	SPC0
4004_6000	6	Wake-Up Unit	WUU0
4004_7000	7	Reserved	—
4004_8000	8	Core Mode Controller	CMC0
4004_9000	9	OS Event Timer	OSTIMER0
4004_A000	10	Low-Power Timer	LPTMR0
4004_B000	11	Low-Power Timer	LPTMR1
4004_C000	12	Real Time Clock Subsystem and Real Time Clock	RTC_SUBSYSTEM0 and RTC0
4004_D000	13	Reserved	—
4004_E000	14	Reserved	—
4004_F000	15	Reserved	—
4005_0000	16	Touch Sensing Interface	TSI0
4005_1000	17	Low Power Comparator	CMP0
4005_2000	18	Low Power Comparator	CMP1
4005_3000	19	Low Power Comparator	CMP2
4005_4000	20	Refer device's Security Reference Manual	—
4005_5000	21	Refer device's Security Reference Manual	—
4005_6000	22	Refer device's Security Reference Manual	—
4005_7000	23	Refer device's Security Reference Manual	—
4005_8000	24	Refer device's Security Reference Manual	—
4005_9000	25	VBAT	VBAT0
4005_A000	26	Refer device's Security Reference Manual	—
4005_B000	27	Error Injection Module	EIM0
4005_C000	28	Error Recording Module	ERM0
4005_D000	29	Interrupt Monitor	INTM0
4005_E000	30	Reserved	—
4005_F000	31	Reserved	—

4.8.2 Peripheral Bridge 1 (PBRG1) memory map

Table 19. Peripheral bridge 1

Base	Slot	Module	Alias
4008_0000	AIPS 1		
4008_0000	0	eDMA 0 Management Page	eDMA_0_MP
4008_1000	1	eDMA 0 CH0 Control and Configuration	eDMA_0_CH0
4008_2000	2	eDMA 0 CH1 Control and Configuration	eDMA_0_CH1
4008_3000	3	eDMA 0 CH2 Control and Configuration	eDMA_0_CH2
4008_4000	4	eDMA 0 CH3 Control and Configuration	eDMA_0_CH3
4008_5000	5	eDMA 0 CH4 Control and Configuration	eDMA_0_CH4
4008_6000	6	eDMA 0 CH5 Control and Configuration	eDMA_0_CH5
4008_7000	7	eDMA 0 CH6 Control and Configuration	eDMA_0_CH6
4008_8000	8	eDMA 0 CH7 Control and Configuration	eDMA_0_CH7
4008_9000	9	eDMA 0 CH8 Control and Configuration	eDMA_0_CH8
4008_A000	10	eDMA 0 CH9 Control and Configuration	eDMA_0_CH9
4008_B000	11	eDMA 0 CH10 Control and Configuration	eDMA_0_CH10
4008_C000	12	eDMA 0 CH11 Control and Configuration	eDMA_0_CH11
4008_D000	13	eDMA 0 CH12 Control and Configuration	eDMA_0_CH12
4008_E000	14	eDMA 0 CH13 Control and Configuration	eDMA_0_CH13
4008_F000	15	eDMA 0 CH14 Control and Configuration	eDMA_0_CH14
4009_0000	16	eDMA 0 CH15 Control and Configuration	eDMA_0_CH15
4009_1000	AHB Peripherals		
4009_1000	17	SCTIMER	SCT0
4009_2000	18	Low-power Flexible Communications Interface	LP_FLEXCOMM0
4009_3000	19	Low-power Flexible Communications Interface	LP_FLEXCOMM1
4009_4000	20	Low-power Flexible Communications Interface	LP_FLEXCOMM2
4009_5000	21	Low-power Flexible Communications Interface	LP_FLEXCOMM3
4009_6000	22	General Purpose Input/Output	GPIO0
4009_7000	23	GPIO 0 (alias 1)	GPIO0_alias1
4009_8000	24	General Purpose Input/Output	GPIO1
4009_9000	25	GPIO 1 (alias 1)	GPIO1_alias1

Table continues on the next page...

Table 19. Peripheral bridge 1 (continued)

Base	Slot	Module	Alias
4009_A000	26	General Purpose Input/Output	GPIO2
4009_B000	27	GPIO 2 (alias 1)	GPIO2_alias1
4009_C000	28	General Purpose Input/Output	GPIO3
4009_D000	29	GPIO 3 (alias 1)	GPIO3_alias1
4009_E000	30	General Purpose Input/Output	GPIO4
4009_F000	31	GPIO 4 (alias 1)	GPIO4_alias1

4.8.3 Peripheral Bridge 2 (PBRG2) memory map

Table 20. Peripheral bridge 2

Base	Slot	Module	Alias
400A_0000	**AIPS 2**		
400A_0000	0	eDMA 1 Management Page	eDMA_1_MP
400A_1000	1	eDMA 1 CH0 Control and Configuration	eDMA_1_CH0
400A_2000	2	eDMA 1 CH1 Control and Configuration	eDMA_1_CH1
400A_3000	3	eDMA 1 CH2 Control and Configuration	eDMA_1_CH2
400A_4000	4	eDMA 1 CH3 Control and Configuration	eDMA_1_CH3
400A_5000	5	eDMA 1 CH4 Control and Configuration	eDMA_1_CH4
400A_6000	6	eDMA 1 CH5 Control and Configuration	eDMA_1_CH5
400A_7000	7	eDMA 1 CH6 Control and Configuration	eDMA_1_CH6
400A_8000	8	eDMA 1 CH7 Control and Configuration	eDMA_1_CH7
400A_9000	9	eDMA 1 CH8 Control and Configuration	eDMA_1_CH8
400A_A000	10	eDMA 1 CH9 Control and Configuration	eDMA_1_CH9
400A_B000	11	eDMA 1 CH10 Control and Configuration	eDMA_1_CH10
400A_C000	12	eDMA 1 CH11 Control and Configuration	eDMA_1_CH11
400A_D000	13	eDMA 1 CH12 Control and Configuration	eDMA_1_CH12
400A_E000	14	eDMA 1 CH13 Control and Configuration	eDMA_1_CH13
400A_F000	15	eDMA 1 CH14 Control and Configuration	eDMA_1_CH14
400B_0000	16	eDMA 1 CH15 Control and Configuration	eDMA_1_CH15
400B_1000	17	Semaphore 2	SEMA42_0
400B_2000	**AHB Peripherals**		
400B_2000	18	Inter-CPU Mailbox	MAILBOX0
400B_3000	19	Refer device's Security Reference Manual	—

Table continues on the next page...

Table 20. Peripheral bridge 2 (continued)

Base	Slot	Module	Alias
400B_4000	20	Low-power Flexible Communications Interface	LP_FLEXCOMM4
400B_5000	21	Low-power Flexible Communications Interface	LP_FLEXCOMM5
400B_6000	22	Low-power Flexible Communications Interface	LP_FLEXCOMM6
400B_7000	23	Low-power Flexible Communications Interface	LP_FLEXCOMM7
400B_8000	24	Low-power Flexible Communications Interface	LP_FLEXCOMM8
400B_9000	25	Low-power Flexible Communications Interface	LP_FLEXCOMM9
400B_A000	26	USB FS RAM	USB0_FS_RAM
400B_B000	27	Refer device's Security Reference Manual	—
400B_C000	28	Refer device's Security Reference Manual	—
400B_D000	29	Refer device's Security Reference Manual	—
400B_E000	30	NPU	NPU0
400B_F000	31	PowerQuad	POWERQUAD0

4.8.4 Peripheral Bridge 3 (PBRG3) memory map

Table 21. Peripheral bridge 3

Base	Slot	Module	Alias
400C_0000	AIPS 3		
400C_0000	0	External Watchdog Monitor	EWM0
400C_1000	1	Performance Monitor - LPCAC	CMX_PERFMON0
400C_2000	2	Performance Monitor - FlexSPI	CMX_PERFMON1
400C_3000	3	Reserved	—
400C_4000	4	Reserved	—
400C_5000	5	Reserved	—
400C_6000	6	Reserved	—
400C_7000	7	Refer device's Security Reference Manual	—
400C_8000	8	Flexible Serial Peripheral Interface	FLEXSPI0
400C_9000	9	Refer device's Security Reference Manual	—

Table continues on the next page...

Table 21. Peripheral bridge 3 (continued)

Base	Slot	Module	Alias
400C_A000	10	Reserved	—
400C_B000	11	Cyclic Redundancy Check	CRC0
400C_C000	12	Refer device's Security Reference Manual	—
400C_D000	13	Reserved	—
400C_E000	14	Enhanced Flex Pulse Width Modulator	PWM0
400C_F000	15	Quadrature Decoder	QDC0
400D_0000	16	Enhanced Flex Pulse Width Modulator	PWM1
400D_1000	17	Quadrature Decoder	QDC1
400D_2000	18	Event Generator	EVTG0
400D_3000	19	Reserved	—
400D_4000	20	FlexCAN with FD	CAN0
400D_8000	21	FlexCAN with FD	CAN1
400D_C000	22	USB FS DCD	USBDCD0
400D_D000	23	USB FS Controller	USBFS0
400D_E000	24	Reserved	—
400D_F000	25	Reserved	—

4.8.5 Peripheral Bridge 4 (PBRG4) memory map

Table 22. Peripheral bridge 4

Base	Slot	Module	Alias
4010_0000	AIPS 4		
4010_0000	0	Ethernet with QoS	ENET0
4010_2000	1	Reserved	—
4010_3000	2	EMVSIM0	EMVSIM0
4010_4000	3	EMVSIM1	EMVSIM1
4010_5000	4	Flexible Input/Output	FLEXIO0
4010_6000	5	Serial Audio Interface	SAI0
4010_7000	6	Serial Audio Interface	SAI1
4010_8000	7	SINC Filter	
4010_9000	8	Ultra Secured Digital Host Controller	uSDHC0
4010_A000	9	USB HS PHY & DCD	USBHS1_PHY
4010_B000	10	USB HS Controller	USBHS1

Table continues on the next page...

Table 22. Peripheral bridge 4 (continued)

Base	Slot	Module	Alias
4010_C000	11	Microphone Interface	MICFIL0
4010_D000	12	Analog-to-Digital Converter	ADC0
4010_E000	13	Analog-to-Digital Converter	ADC1
4010_F000	14	12-bit Digital-to-Analog Converter	DAC0
4011_0000	15	Operational Amplifier	OPAMP0
4011_1000	16	Voltage Reference	VREF0
4011_2000	17	12-bit Digital-to-Analog Converter	DAC1
4011_3000	18	Amplifier	OPAMP1
4011_4000	19	14-bit Digital-to-Analog Converter	DAC2
4011_5000	20	Operational Amplifier	OPAMP2
4011_6000	21	Port Control	PORT0
4011_7000	22	Port Control	PORT1
4011_8000	23	Port Control	PORT2
4011_9000	24	Port Control	PORT3
4011_A000	25	Port Control	PORT4
4011_B000	26	Reserved	—
4011_C000	27	Reserved	—
4011_D000	28	Reserved	—
4011_E000	29	Reserved	—
4011_F000	30	Reserved	—
4012_0000	AHB Peripherals		
4012_0000	32	Secure AHB Controller	AHBSC
4012_1000	33	Secure AHB Controller Alias 1	AHBSC_alias1
4012_2000	34	Secure AHB Controller Alias 2	AHBSC_alias2
4012_3000	35	Secure AHB Controller Alias 3	AHBSC_alias3
4012_4000		Reserved	—

Chapter 5
Low-Power Cache Controller (LPCAC)

5.1 Chip-specific LPCAC information

Table 23. Reference links to related information

Topic	Related module	Reference
Full description	LPCAC	LPCAC
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

See SYSCON LPCAC Control (LPCAC_CTRL) description for the LPCAC controls.

5.1.1 Module instances

This device has one instance of the LPCAC module.

5.1.2 LPCAC cacheable memory range

The LPCAC module has no registers. All configuration for the LPCAC module is handled through the SYSCON's LPCAC_CTRL register.

The 16 KB LPCAC is connected to the Code bus of the primary CM33 core. Contents of this cache are only visible to the CM33 core. The cacheable memory range are shown in table below. For details, refer to System memory map.

Table 24. LPCAC cacheable memory range

Start address	End address	TZ-M State	Description	Size	Cached
0000_0000	001F_FFFF	Non-Secure	Program flash **Note:** Program flash has two flash memory arrays/banks of up to 1 MB each.	2 MB	LPCAC
0800_0000	0FFF_FFFF	Non-Secure	Flexible Serial Peripheral Interface (FlexSPI) **Note:** The 128 MB FlexSPI at address 0x0800_0000 is aliased to FlexSPI at address 0x8000_0000 ~0x87FF_FFFF.	128 MB	LPCAC/ CACHE64
1000_0000	101F_FFFF	Secure	Program flash	2 MB	LPCAC
1800_0000	1FFF_FFFF	Secure	Flexible Serial Peripheral Interface (FlexSPI)	128 MB	LPCAC/ CACHE64

5.1.3 LPCAC reset

A system reset would reset the LPCAC on this device. LPCAC is disabled after the reset.

5.1.4 Parity error check

LPCAC supports parity checking on its RAM. Any parity errors that are detected are reported through the Error Recording Module (ERM). See the ERM chapter.

5.2 Overview

LPCAC provides low-latency access to instructions or data. This decouples processor performance from system memory performance, increasing bus availability for other modules and improving system performance.

LPCAC has a 32-bit data path AMBA-AHB input bus and a 32-bit data path AMBA-AHB output bus.

5.2.1 Block diagram

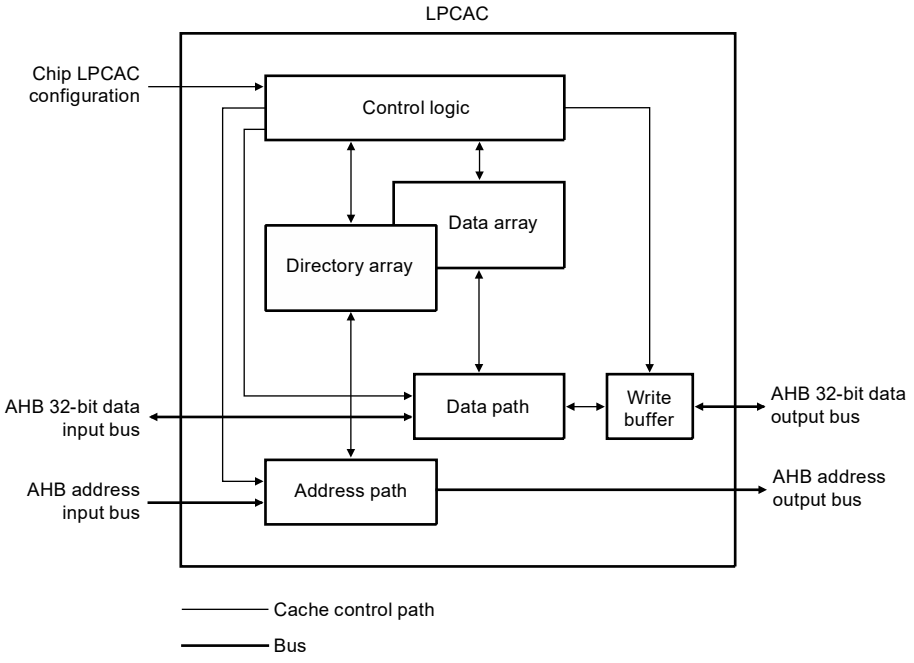


Figure 5. Block diagram

5.2.2 Features

LPCAC supports the following features:

- Nonblocking and write-through cache mode
- 16 KB total cache size
- Cache organization as an 8-way, 8-set-associative design based on 256-byte superpages
- Each way or superpage contains up to sixteen sequential 16-byte pages
- Cache supports optional odd parity with one bit of odd parity per 4 bytes of cache data.
  - Cache can recover and scrub data parity errors by forcing miss on parity error and reloading data

5.3 Functional description

This section provides further information on the LPCAC module operation.

A reset signal clears and enables this module.

LPCAC examines every valid AHB input access. If the access hits the cache memory region and the cache is enabled, the access goes to the cache portion of the module. If the access is not cacheable, the access is passed directly to the AHB output bus.

The LPCAC cache has a 256-byte line subdivided into sixteen 16-byte sublines:

- Cache read accesses that are cache misses perform an aligned 16-byte subline-size burst cache read miss to the module's AHB output bus. Then the cache loads the needed data in the cache’s data storage. The cache miss uses a 4-beat (32 bits per beat), wrapped burst bus access to fetch the cache miss data.
- Cache read accesses that are cache hits return the desired read data from the cache.
- Cache write accesses that are cache misses perform the desired write operation only to the module's AHB output bus (for write-through mode accesses, LPCAC has a no allocation on write-miss policy).
- Cache write accesses that are cache hits perform the desired write operation to the cache and the module's AHB output bus.

LPCAC has a one entry write buffer. When LPCAC is enabled and available, cache write accesses with a bufferable attribute use the buffer, which allows write from the processor to receive an immediate (zero wait state) bus termination.

5.3.1 Cache functional description

The LPCAC cache is an 8-way, 8-set-associative, 256-byte per line write-through design. It supports a total cache capacity of 16 KB for a 32-bit wide cache miss data path.

5.3.1.1 Cache controls

This module has controls to enable, disable, and clear the cache.

The cache control inputs are as follows:

- clr_lpcac: clear lpcac
- dis_lpcac: disable lpcac
- dis_lpcac_wtbf: disable write buffer
- lim_lpcac_wtbf: limit write buffer
- frc_no_alloc: force no allocation
- parity_miss_en: enable parity, miss on parity error
- parity_fault_en: enable parity error reporting

See chip-specific section for more information on how these signals are connected or controlled on a given device.

5.3.2 Clocks

The core clock domain defines the module's clock domain.

5.3.3 Reset

A reset signal clears and enables this module. See the chip-integration information for the resets that affect LPCAC.

5.3.4 Interrupts

This module has no interrupts.

5.4 Signal descriptions

This module has no external signals.

5.5 Memory regions and input control description

5.5.1 Memory regions

LPCAC decodes cacheable address regions. The cache memory region may be composed of one or more disjointed memory regions and the total size may be larger than the cache storage.

Any address not in the cacheable memory region is in the pass-through memory region. All accesses to the pass-through memory regions are non-cacheable. For all accesses to the cacheable memory regions, the access is cacheable if the attributes of the access indicate that it can be cached, else the access is non-cacheable.

See the System Memory Map for cache memory range information.

5.5.2 Input controls

The LPCAC does not have a module-resident programming model. An external block supplies the needed LPCAC control inputs.

Table 25. Control inputs for LPCAC

Function	Control input	Description
clear lpcac	clr_lpcac	One cycle active-high pulse clears the lpcac.
disable lpcac	dis_lpcac	• 0 = lpcac enabled (reset configuration) • 1 = lpcac disabled, all cacheable accesses pass from LPCAC input to output bus
disable write buffer	dis_lpcac_wtbf	• 0 = write buffer enabled • 1 = write buffer disabled
limit write buffer	lim_lpcac_wtbf	• 0 = If write buffer is enabled, buffer all writes to spaces that are bufferable • 1 = If write buffer is enabled, buffer all writes to spaces that are both bufferable and cacheable
force no allocation	frc_no_alloc	When frc_no_alloc is asserted and the cache is enabled, all accesses to the cache search the cache. If they hit a valid entry that was loaded before frc_no_alloc was asserted, they operate normally. That is, read hits return cache data without going through the cache and write hits update cache data and write through the cache. If they miss, they bypass the cache (that is, even though a read access is to a cacheable space, it does not allocate on a cache miss). This control input is useful for debug. If there is a software breakpoint, halting the processor. Then, using the debug port, cacheable memory and other address spaces are accessed. These debug accesses go through the processor, through the cache, and to the rest of the system. The idea is during the debug accesses the cache is placed on frc_no_alloc mode. In this way, debug access through the core and then through the cache do not disturb the state of the cache. Before restarting the processor, frc_no_alloc mode is negated. When the processor restarts, the cache state is the same or at least very similar as the cache state when the breakpoint was hit. • 0 = normal allocation on cache miss • 1 = no allocation on cache miss

Table continues on the next page...

Table 25. Control inputs for LPCAC (continued)

Function	Control input	Description
parity miss enable	parity_miss_en	Enables miss on parity to scrub the parity error and to reload the data line that had the parity error. • 0 = do not take a miss on a parity error • 1 = force a miss on a parity error
parity enable	parity_fault_en	Enables and disables parity error report. • 0 = disable • 1 = enable

Chapter 6
CACHE64 Controller (CACHE64_CTRL)

6.1 Chip-specific CACHE64_CTRL information

Table 26. Reference links to related information

Topic	Related module	Reference
Full description	CACHE64_CTRL	CACHE64_CTRL
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	NA

NOTE

The CACHE64_CTRL module works in conjunction with the CACHE64_POLSEL module. The CACHE64_POLSEL determines the cache policy to be applied for a given AHB address and passes this information to the CACHE64_CTRL.

6.1.1 Module instances

This device has one instance of the CACHE64 controller module, CACHE64_CTRL0.

6.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by non-secure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

6.1.3 FlexSPI regions mapping to CACHE64_CTRL

The CACHE64_CTRL module is used to cache FlexSPI accesses. There are multiple FlexSPI regions in the system memory map, but these are remapped to create a single contiguous 512 Mbyte region (0x8000_0000-0x9FFF_FFFF), where accesses to some system regions are aliased into a single region for the cache. The table below shows how the FlexSPI system regions are mapped to the CACHE64_CTRL:

Table 27. FlexSPI regions mapping to CACHE64_CTRL

System start address (hex)	System End address (hex)	Size	CACHE64_CTRL start address	CACHE64_CTRL end address	Access
0x0800_0000	0x0FFF_FFFF	128 MB	0x8000_0000	0x87FF_FFFF	Non-secure
0x1800_0000	0x1FFF_FFFF	128 MB	0x8000_0000	0x87FF_FFFF	Secure
0x8000_0000	0x8FFF_FFFF	256 MB	0x8000_0000	0x8FFF_FFFF	Non-secure
0x9000_0000	0x9FFF_FFFF	256 MB	0x8000_0000	0x8FFF_FFFF	Secure
0xA000_0000	0xAFFF_FFFF	256 MB	0x9000_0000	0x9FFF_FFFF	Non-secure
0xB000_0000	0xBFFF_FFFF	256 MB	0x9000_0000	0x9FFF_FFFF	Secure

6.2 Overview

A cache is a block of high-speed memory locations that contains address information (commonly known as a tag) and its associated data. It decreases the average time of a memory access. CACHE64_CTRL includes cache memory controller (CACHE64), which is a general purpose AMBA AHB bus protocol cache.

6.2.1 Block diagram

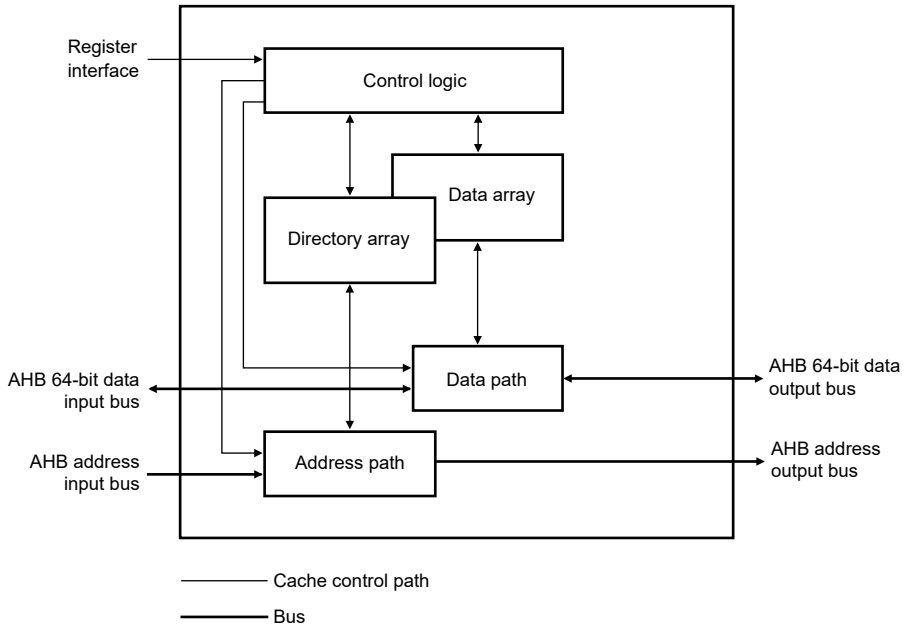


Figure 6. Block diagram

6.2.2 Features

CACHE64_CTRL is designed to work with AMBA AHB bus systems. It includes:

- An input (Controller) AHB bus that you can access using Cache mode attributes for cacheable copyback, cacheable write-through, or noncacheable mode operation and write bufferable mode operation on an access-by-access basis.
- An output (Target) 64-bit data path AHB bus for downstream cache read miss, cache write-through write, and noncacheable accesses.
- A peripheral interface for accessing the cache's program model.

6.3 Functional description

A cache operates on two principles of locality:

- Spatial locality — Access to one location possibly follows an access from adjacent locations (for example, sequential instruction execution or usage of a data structure).
- Temporal locality — Access to an area of memory possibly repeats within a short period of time (for example, execution of a code loop).

To minimize the quantity of control information stored, the spatial locality property groups several locations together, under the same tag. This logical block is known as a cache line.

You achieve temporal locality by retaining recently accessed lines in the cache.

When data loads into a cache, access times for subsequent loads and stores are reduced, resulting in overall performance benefits. An access to information already in a cache is known as a cache hit, and other accesses are known as cache misses.

Normally, caches are self-managing, with the updates occurring automatically. Whenever the bus controller wants to access a cacheable location, the cache is checked. If the access is a cache hit, the access occurs immediately. Otherwise, a location is allocated, and the cache line is loaded from memory. Different cache topologies and access policies are possible. However, they must comply with the memory coherency model of the underlying architecture.

Caches introduce a number of potential problems, mainly because of:

- Memory accesses occurring at times other than when the programmer would normally expect them.
- The existence of multiple physical locations where a data item can be held.

6.3.1 Submodules

- Cache_apb — This sub-module is responsible for CACHE64_CTRL configuration of related registers.
- Cache_cam — This sub-module contains address information (commonly known as a tag) and its associated data. This sub-module asserts TAG parity fault.
- Cache_pipe — This sub-module is responsible for AHB bus access.

6.3.2 Operations

6.3.2.1 Cache configuration

CACHE64_CTRL receives manager bus requests on the manager (M) bus.

You can access the cache programming model via the cache controller's AMBA AHB bus.

CACHE64_CTRL then processes the cacheable accesses as needed, when bypassing the noncacheable, cache write-through, cache miss, and cache maintenance accesses to its subordinate bus.

The cache on this chip has a 2-way set associative cache structure with a total size of 16 KB. The cache has a 32-bit address, 64-bit data paths, and a 32-byte line size. The cache tags and data storage use single-port, synchronous RAMs.

For this 16 KB cache, each cache TAG function uses two 256 x 21-bit RAM arrays, and the cache DATA function uses two 1024 x 64-bit RAM arrays. The cache TAG entries store 19 bits of the upper address as well as a modified and valid bit per cache line. The cache DATA entries store eight bytes of code or data.

All normal cache accesses use physical addresses. This leads to the use of the following cache address:

CACHE - 16 KB size = (256 sets) x (32-byte lines) x (2-way set associative)

Table 28. Tag cache address use

Tag cache address use	Cache
Tag hit address range	Address[31:13]
Tag set select address range	Address[12:5] used to select 1 of 256 sets
Not used	Address[4:0]

Table 29. Data cache address use

Data cache address use	System cache
Not used	Address[31:13]
Data set select address range	Address[12:5] used to select one of 256 sets
64-bit word select	Address[4:3] used to select one of four 64-bit words within a set
Byte select	Address[2:0] used to select the byte within the 64-bit word

6.3.2.2 Cache control

The cache is disabled at reset. Cache tag and data arrays are not cleared at reset. Therefore, to enable and configure the cache, you must clear the cache commands and initialize the required tag array.

6.3.2.2.1 Cache set commands

The cache set commands may operate on one of the following:

- All of way 0
- All of way 1
- All of both ways (complete cache)

Cache set commands are specified using the upper bits in Cache Control (CCR). Cache set commands perform their operation on the cache independent of CCR[ENCACHE].

You initiate a cache set command by writing 1 to CCR[GO]. This field also acts as a busy bit for set commands. The field remains 1 when the command is active, and the module writes 0 to it after the set command completes.

Supported cache set commands are provided in Table 30. Following are the set commands:

- Invalidate — Unconditionally clears valid and modify bits of a cache entry.
- Push — Pushes a cache entry if it is valid and modified, then clears the modify bit. If the entry is not valid or not modified, leave it as is.
- Clear — Pushes a cache entry if it is valid and modified, then clears the valid and modify bits. If the entry is not valid or not modified, clear the valid bit.

Table 30. Cache set commands

Cache Control (CCR)[27:24]				Command
PUSHW1	INVW1	PUSHW0	INVW0	
0	0	0	0	No operation (NOP)
0	0	0	1	Invalidate all of way 0
0	0	1	0	Push all of way 0
0	0	1	1	Clear all of way 0
0	1	0	0	Invalidate all of way 1
0	1	0	1	Invalidate all of way 1 and way 0 (invalidate complete cache)
0	1	1	0	Invalidate all of way 1; push all of way 0
0	1	1	1	Invalidate all of way 1; clear all of way 0
1	0	0	0	Push all of way 1
1	0	0	1	Push all of way 1; invalidate all of way 0
1	0	1	0	Push all of way 1 and way 0 (push complete cache)
1	0	1	1	Push all of way 1; clear all of way 0
1	1	0	0	Clear all of way 1
1	1	0	1	Clear all of way 1; invalidate all of way 0
1	1	1	0	Clear all of way 1; push all of way 0
1	1	1	1	Clear all of way 1 and way 0 (clear complete cache)

After a reset, complete an invalidate cache command before using the cache. It is possible to combine the cache invalidate command with the cache enable. That is, writing 8500_0003h to Cache Control (CCR) invalidates the cache and enables the cache and write buffer.

6.3.2.2.2 Cache line commands

Cache line commands operate on a single line in the cache at a time. You can perform these commands using a physical or cache address that consists of a set address and a way select. The line command acts on the specified cache line.

Cache line commands with physical addresses first search both ways of the cache set specified by the following physical address bits. If they hit, the commands perform their action on the hit way: For cache- [12:5].

You specify cache line commands using the upper bits in Cache Line Control (CLCR). Cache line commands perform their operation on the cache independent of CCR[ENCACHE]. Using a cache address, the command can be completely specified using Cache Line Control (CLCR). Using a physical address, the command must also use Cache Search Address (CSAR) to specify the physical address.

You can initiate a line cache command by writing 1 to CLCR[LGO] or CSAR[LGO]. This field also acts as a busy bit for line commands. The field remains 1 when the command is active and CACHE64_CTRL writes 0 to it after the set command completes.

Table 31. Cache line commands

Cache Line Control (CLCR)[27:24]			Command
CLCR[LACC]	CLCR[LADSEL]	CLCR[LCMD]	
0	0	00	Search by cache address and way
0	0	01	Invalidate by cache address and way
0	0	10	Push by cache address and way
0	0	11	Clear by cache address and way
0	1	00	Search by physical address
0	1	01	Invalidate by physical address
0	1	10	Push by physical address
0	1	11	Clear by physical address
1	0	00	Write by cache address and way
1	0	01	NOP
1	0	10	NOP
1	0	11	NOP
1	1	*xx*	NOP

6.3.2.2.2.1 Executing a series of line commands using cache addresses

You can perform a series of line commands with incremental cache addresses by writing to Cache Line Control (CLCR):

1. Place the command in Cache Line Control (CLCR)[27:24].
2. Write 1 to CLCR[WSEL] and CLCR[TDSEL] as needed.
3. Place the cache address in CLCR[CACHEADDR].
4. Write 1 to CLCR[LGO].
5. Wait for the command to complete (CLCR[LGO] becomes 0).
6. Increment the cache address (at bit 3 to step through data or at bit 5 to step through lines).

7. Write 1 to CLCR[LGO].
8. Wait for the command to complete (CLCR[LGO] becomes 0).
9. Repeat steps 5 - 8 as needed.

6.3.2.2.2.2 Executing a series of line commands using physical addresses

You can perform a series of line commands with incremental physical addresses using the following steps:

1. Write to Cache Line Control (CLCR):
   a. Place the command in Cache Line Control (CLCR) [27:24].
   b. Write 1 to CLCR[TDSEL].
2. Place the physical address in CSAR[PHYADDR] and write 1 to CSAR[LGO].
3. Wait for the command to complete (CLCR[LGO] becomes 0).
4. Increment the physical address (at bit 3 to step through data or at bit 5 to step through lines).
5. Write 1 to CLCR[LGO].
6. Wait for the command to complete (CLCR[LGO] becomes 0).
7. Repeat steps 3 - 6 as needed.

Cache Line Control (CLCR) and Cache Search Address (CSAR) both contain a field called LGO, and this field has the same functionality. Therefore, you can complete the above steps in a single write to Cache Search Address (CSAR).

6.3.2.2.2.3 Line command results

At completion of a line command, Cache Line Control (CLCR) contains information on the initial state of the line targeted by the command. For line commands with cache addresses, read this information before performing a line command action from the targeted cache line. For line commands with physical addresses, read this information on a hit before the line command action is performed from the hit cache line. In the cache, for line commands with a physical address miss, CLCR[LCIVB] becomes 0, and no line command action is performed. In general, if CLCR[LCIVB] becomes 0, the targeted line is invalid at the start of the line command and no line operation is performed.

Table 32. Line command results

Cache Line Control (CLCR)[22:20]			For cache address commands	For physical address commands
CLCR[LCWAY]	CLCR[LCIMB]	CLCR[LCIVB]		
0	0	0	Way 0 line is invalid	No hit
0	0	1	Way 0 valid and not modified	Way 0 valid and not modified
0	1	0	Way 0 line is invalid	No hit
0	1	1	Way 0 valid and modified	Way 0 valid and modified
1	0	0	Way 1 line is invalid	No hit
1	0	1	Way 1 valid and not modified	Way 1 valid and not modified
1	1	0	Way 1 line is invalid	No hit
1	1	1	Way 1 valid and modified	Way 1 valid and modified

On completion of a line command other than a write, Cache Read/Write Value (CCVR) contains information on the initial state of the line tag or data targeted by the command. For line commands, CLCR[TDSEL] selects between tag and data. If the line command used a physical address and missed, the data is don't care. For write commands, Cache Read/Write Value (CCVR) holds the write data.

The cache does not have line lock capability.

6.3.3 Modes of operation

Table 33. Modes of operation

Mode	Description
Write-Through	Access to address spaces with this cache mode are cacheable: • If all cacheable spaces are read-only spaces, then cache contains read-only data and all writes to the cache cause a fault. See the chip-specific cacheable space information. • A read miss on the input bus causes a line read on the output bus of a 32-byte-aligned memory address that contains the desired address. A read miss data loads into the cache and marks as valid and not modified. • A read hit to a valid cache location returns data from the cache with no output bus accesses. • A write miss bypasses the cache and writes to the output bus (no allocation on write-miss policy for Write-Through mode spaces). • A write hit updates the cache hit line with the write data and writes to the output bus. • The caches are bus-controller-local and do not support hardware cache coherency. If the bus controller accesses write-through regions and another bus controller that can access the same target memory, without going through the cache (such as DMA), must update these regions, you must first perform explicit cache clears to ensure all modified cache lines update their associated memories, before they are modified by external controllers and subsequent cache bus controller accesses get the updated memory.
Write-Back	Access to address spaces with this cache mode are cacheable: • A read miss on the input bus causes a line read on the output bus of a 32-byte-aligned memory address that contains the desired address. This miss data loads into the cache and marks it as valid and not modified. • A read hit to a valid cache location returns data from the cache with no output bus access. • A write miss performs a read-to-write (executes write-miss policy for Write-Back mode spaces). You perform a line read on the output bus of a 32-byte-aligned memory address containing the desired write address. This miss data is loaded into the cache and marked as valid and modified. Then the write data updates the appropriate cache data locations.

Table continues on the next page...

Table 33. Modes of operation (continued)

Mode	Description
	• A write hit updates the cache hit line with the write data and marks the line as modified. • Caches do not support hardware cache coherency. If the cache's bus controller has written to write-back regions, and another bus controller that can access the same target memory without going through the cache must see these updates, you must perform explicit cache pushes to any needed cache memory range to ensure that all modified cache lines update their associated memories before being this controller reads them. Likewise, if the cache's bus controller has accessed write-back regions, and another bus controller that can access the same target memory without going through the cache must update these regions, you must first perform explicit cache clears to any needed cache memory range to ensure that all modified cache lines update their associated memories before this controller modifies them, and subsequent cache controller bus accesses get the updated memory.
Noncacheable	Access to address spaces with this cache mode are not cacheable. These accesses bypass the cache and access the output bus.

6.3.4 Clocking

Caches use the AHB manager bus clock.

6.3.5 Reset

Caches are disabled at reset. However, cache tag and data arrays are not cleared at reset. See Initialization and Cache control for more information regarding cache initialization and configuration.

6.3.6 Interrupts

CACHE64_CTRL has no interrupts.

6.4 External signals

There are no external signals for CACHE64_CTRL.

6.5 Initialization

Perform this procedure to initialize cache:

1. Write 1 to CCR[INVW1] and CCR[INVW0].
2. Write 1 to CCR[GO].
3. Write 1 to CCR[ENWRBUF] and CCR[ENCACHE].

NOTE

Steps 1, 2 and 3 can be done simultaneously. That is, writing 8500_0003h to Cache Control (CCR) invalidates the cache and enables the cache and write buffer.

6.6 Memory map and registers

The cache programmer's model provides a variety of registers for configuring and controlling the cache, as well as indirect access paths to all cache tag and data storage.

NOTE

The cache registers are accessible in Supervisor mode only.

6.6.1 CACHE64_CTRL register descriptions

6.6.1.1 CACHE64_CTRL memory map

CACHE64_CTRL0 base address: 4001_B000h

Offset	Register	Width (In bits)	Access	Reset value
800h	Cache Control (CCR)	32	RW	0000_0000h
804h	Cache Line Control (CLCR)	32	RW	0000_0000h
808h	Cache Search Address (CSAR)	32	RW	0000_0000h
80Ch	Cache Read/Write Value (CCVR)	32	RW	0000_0000h

6.6.1.2 Cache Control (CCR)

Offset

Register	Offset
CCR	800h

Function

Enables the cache and write buffer. This register also contains the fields used for commands specified in Cache set commands.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GO	0			PUSHW1	INVW1	PUSHW0	INVW0	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												FRCNOAL...	FRCWT	ENWRBUF	ENCACHE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 GO	Initiate Cache Command Initiates the cache command that CCR[INVW0], CCR[PUSHW0], CCR[INVW1], and CCR[PUSHW1] indicate, if you write 1 to this field. Reading this field indicates whether a cache command is active. **NOTE** This field remains 1 until the cache command completes. Writing 0 has no effect. 0b - Write: no effect; Read: no cache command active 1b - Write: initiates cache command; Read: cache command active
30-28 —	Reserved
27 PUSHW1	Push Way 1 Specifies the cache command to execute the next time CCR[GO] becomes 1. 0b - No operation 1b - Push all modified lines in way 1
26 INVW1	Invalidate Way 1 Specifies the cache command to execute the next time CCR[GO] becomes 1. **NOTE** If CCR[PUSHW1] and CCR[INVW1] are 1, then after you write 1 to CCR[GO], this field pushes all modified lines in way 1 and invalidates all lines in way 1 (clear way 1). 0b - No operation 1b - Invalidates all lines in way 1
25 PUSHW0	Push Way 0 Specifies the cache command to execute the next time CCR[GO] becomes 1. 0b - No operation 1b - Push all modified lines in way 0
24 INVW0	Invalidate Way 0 Specifies the cache command to execute the next time CCR[GO] becomes 1. **NOTE** If CCR[PUSHW0] and CCR[INVW0] are 1, then after you write 1 to CCR[GO], this field pushes all modified lines in way 0 and invalidates all lines in way 0 (clear way 0). 0b - No operation 1b - Invalidates all lines in way 0
23-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3 FRCNOALLC	Forces No Allocation On Cache Misses 0b - Allocation on cache misses 1b - Forces no allocation on cache misses (FRCWT must be asserted)
2 FRCWT	Force Write Through Mode Specifies whether all cacheable spaces to write through are forced. 0b - Does not force 1b - Force
1 ENWRBUF	Enable Write Buffer Enables the write buffer. 0b - Disables 1b - Enables
0 ENCACHE	Cache Enable Enables the cache. 0b - Disables 1b - Enables

6.6.1.3 Cache Line Control (CLCR)

Offset

Register	Offset
CLCR	804h

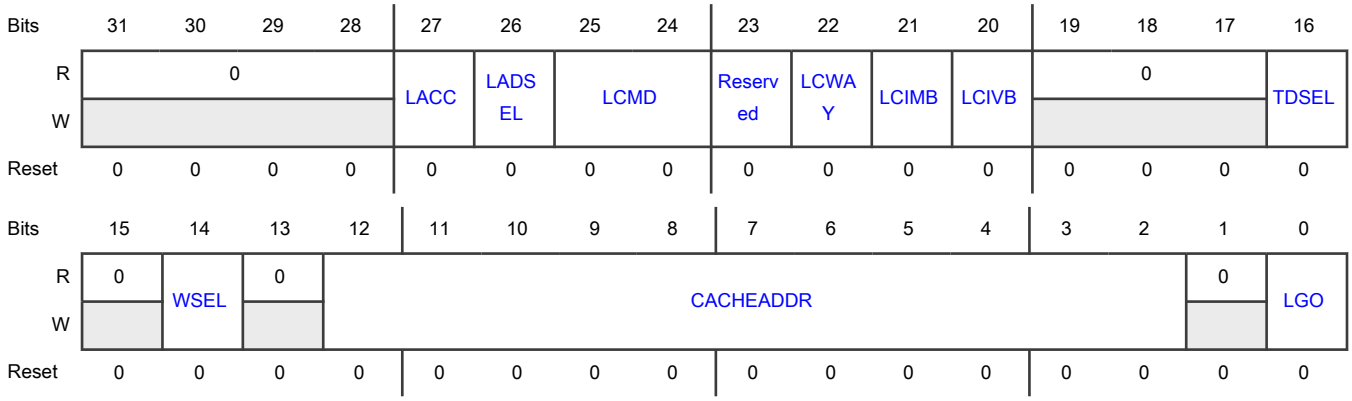
Function

Defines the commands specified in Cache line commands that you must perform using a specific cache line address or physical address.

If you specify the physical address, you can search cache in both the ways, and the command only performs on the way that hits.

You can write to CLCR[LCWAY], CLCR[LCIMB], and CLCR[LCIVB]; however, the values that you write to these fields have no functional purpose.

Diagram



Fields

Field	Function
31-28 —	Reserved
27 LACC	Line Access Type Specifies the operation that you must perform (read or write) after a hit, when performing a search command. **NOTE** Only used for search commands (CLCR[LCMD] = 00). 0b - Read 1b - Write
26 LADSEL	Line Address Select Specifies the address (cache or physical). When using the cache address (specified in CLCR[WSEL] and CLCR[CACHEADDR]), the way must also be specified in CLCR[WSEL]. When using the physical address (specified in Cache Search Address (CSAR)), both ways are searched and the command is performed only after a hit. 0b - Cache 1b - Physical
25-24 LCMD	Line Command Performs one of the four commands after a cache hit. CLCR[LACC] determines whether the operation is a read or write. 00b - Search and read or write 01b - Invalidate 10b - Push 11b - Clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 —	Reserved
22 LCWAY	Line Command Way Indicates the way that the line command uses. Applies only if CLCR[LCIVB] = 1. 0b - Way 0 1b - Way 1
21 LCIMB	Line Command Initial Modified Bit Shows the initial state of the modified bit. If this field = 0, and the command uses cache address and way, this field shows the initial state of the modified bit, which is 0. If the command uses physical address and a hit, this field shows the initial state of the modified bit, which is 0. If the command uses physical address and is a miss, this field reads 0. If this field = 1, and the command uses cache address and way, this field shows the initial state of the modified bit, which is 1. If the command uses physical address and way, this field shows the initial state of the modified bit, which is 1. If a miss, this field reads 0. 0b - Initial state 0 1b - Initial state 1
20 LCIVB	Line Command Initial Valid Bit Shows the initial state of the valid bit. If this field = 0, and the command uses cache address and way, this field shows the initial state of the valid bit, which is 0. If the command uses physical address and a hit, this field shows the initial state of the valid bit, which is 0. If the command uses physical address and is a miss, this field reads 0. If this field = 1, and the command uses cache address and way, this field shows the initial state of the valid bit, which is 1. If the command uses physical address and way, this field shows the initial state of the modified bit, which is 1. If a miss, this field reads 0. 0b - Initial state 0 1b - Initial state 1
19-17 —	Reserved
16 TDSEL	Tag Or Data Select Selects tag or data for search and read or write commands. 0b - Data 1b - Tag
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14 WSEL	Way Select Selects the way for line commands. 0b - Way 0 1b - Way 1
13 —	Reserved
12-2 CACHEADDR	Cache Address Represents bits [12:2] of the cache address: CLCR[12:5] accesses the tag arrays. CLCR[12:3] accesses the data arrays. CLCR[2] specifies which half of the 64 bits of data to place in Cache Read/Write Value (CCVR).
1 —	Reserved
0 LGO	Initiate Cache Line Command Initiates the cache command that CLCR[LCMD], CLCR[LADSEL], and CLCR[LACC] indicate, if you write 1 to this field. Reading this field indicates whether a cache command is active. **NOTE** • This field remains 1 until the command completes. Writing 0 has no effect. • This field is shared with CSAR[LGO]. You can write either register address bit[0] interchangeably. This saves cache writes (helps performance), when performing cache operations. Cache line operations start when you write to Cache Line Control (CLCR) with CLCR[LGO] being 1 or the cache search address can be changed and a search is started by writing to Cache Search Address (CSAR) with CSAR[LGO] being 1. 0b - Write: no effect; Read: no line command active 1b - Write: initiate line command; Read: line command active

6.6.1.4 Cache Search Address (CSAR)

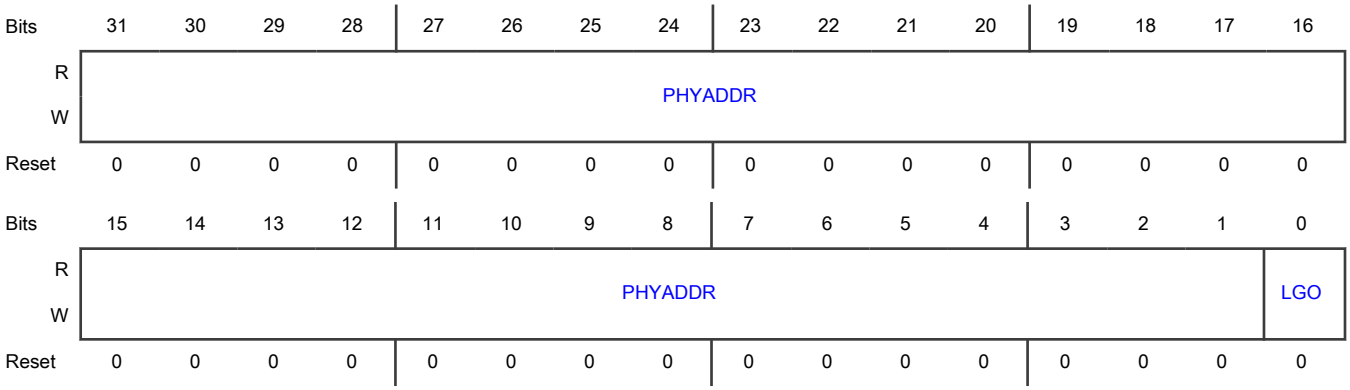
Offset

Register	Offset
CSAR	808h

Function

Defines the physical address for cache line commands, when CLCR[LADSEL] = 1.

Diagram



Fields

Field	Function
31-1 PHYADDR	Physical Address Represents bits [31:1] of the system address: CSAR[31:13] is used for tag compare. CSAR[12:5] accesses the tag arrays. CSAR[12:3] accesses the data arrays. CSAR[2] specifies which half of the 64 bits of data to place in Cache Read/Write Value (CCVR). CSAR[1] is not used. Writes to this bit are allowed, and the read value is the previously written value. This bit is not functionally used.
0 LGO	Initiate Cache Line Command Initiates the cache command that CSAR[PHYADDR] indicates, if you write 1 to this field. Reading this field indicates whether a cache command is active. **NOTE** • This field remains 1 until the command completes. Writing 0 has no effect. • This field is shared with CLCR[LGO]. 0b - Write: no effect; Read: no line command active 1b - Write: initiate line command; Read: line command active

6.6.1.5 Cache Read/Write Value (CCVR)

Offset

Register	Offset
CCVR	80Ch

Function

Sources write data or return read data when a search line command is requested (CLCR[LCMD] = 00). CLCR[LACC] determines whether the access is a read or write, and CLCR[TDSEL] determines whether the access is to the data or tag arrays.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DATA	Cache Read/Write Data Reads or writes data during tag search and data search. For tag search, read or write: • CCVR[31:13] for tag array R/W value. • CCVR[12:5] for tag set address on reads; unused on writes. • CCVR[4:2] are reserved. • CCVR[1] for tag modify bit. • CCVR[0] for tag valid bit. For data search, read or write: • CCVR[31:0] for data array R/W value.

Chapter 7
CACHE64 Policy Select (CACHE64_POLSEL)

7.1 Chip-specific CACHE64_POLSEL information

Table 34. Reference links to related information

Topic	Related module	Reference
Full description	CACHE64_POLSEL	CACHE64_POLSEL
System memory map		Memory map
Clocking		Clock distribution

7.1.1 Module instances

This device has one instance of the CACHE64_POLSEL module, CACHE64_POLSEL0.

7.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

7.1.3 FlexSPI regions mapping to CACHE64_POLSEL

The CACHE64_POLSEL block is responsible for defining the memory regions and cache polices to use for each region within the CACHE64 controller. Because the CACHE64 is integrated at the FlexSPI slave port, the cacheable regions are the FlexSPI regions in the system memory map. The upper address lines are ignored when setting cache policies, so that the three regions supported by CACHE64_POLSEL are applied across all of the FlexSPI address regions.

There are multiple FlexSPI regions in the system memory map, but these are remapped to create a single contiguous 512 Mbyte region (0x8000_0000-0x9FFF_FFFF), where accesses to some system regions are aliased into a single region for the cache. The table below shows how the FlexSPI system regions are mapped to the CACHE64_POLSEL:

Table 35. FlexSPI regions mapping to CACHE64_POLSEL

System start address (hex)	System End address (hex)	Size	CACHE64_POLSEL start address	CACHE64_POLSEL end address	Access
0x0800_0000	0x0FFF_FFFF	128 MB	0x8000_0000	0x87FF_FFFF	Non-secure
0x1800_0000	0x1FFF_FFFF	128 MB	0x8000_0000	0x87FF_FFFF	Secure
0x8000_0000	0x8FFF_FFFF	256 MB	0x8000_0000	0x8FFF_FFFF	Non-secure
0x9000_0000	0x9FFF_FFFF	256 MB	0x8000_0000	0x8FFF_FFFF	Secure
0xA000_0000	0xAFFF_FFFF	256 MB	0x9000_0000	0x9FFF_FFFF	Non-secure
0xB000_0000	0xBFFF_FFFF	256 MB	0x9000_0000	0x9FFF_FFFF	Secure

7.2 Overview

CACHE64_POLSEL monitors incoming AHB addresses to define the policy that the cache controller (CACHE64_CTRL) uses. Both CACHE64_POLSEL and the cache controller work in tandem with each other.

7.2.1 Features

- Provides cache policy information to the cache controller.
- Supports up to three regions, each with a programmable cache policy:
  - Each region can be independently designated as noncache, write-through cache, and write-back cache.
  - User-defined boundaries are supported between the three regions with 1 KB granularity.

7.3 Functional description

CACHE64_POLSEL permits dividing the memory space that the cache controller accesses into three contiguous regions and applies one of these three cache policies to each region:

- First region (region 0) starts at the bottom (lowest address) in the target address space and goes up to the top of the 1 KB boundary designated in REG0_TOP.
- Second region (region 1) starts at the next address and extends to the top of the boundary designated in REG1_TOP.
- Third region starts at the address after that and continues to the top of the memory space accessed via the cache controller.

7.3.1 Region mapping

CACHE64_POLSEL controls the cache policy for memory between the base address and end address. Within the CACHE64_POLSEL memory area, you can define up to three contiguous memory regions, with each region having independently programmable cache policy settings.

As shown in the following figure, REG0_TOP and REG1_TOP control the boundaries between the regions. These registers provide bits 26-11 of the address and the lower 10 bits are always 1s. For example, if REG0_TOP is 0001_0000h, then region 0 ends at BASE_ADDRESS + 0001_03FFh. Region 1 starts at 0001_0400h.

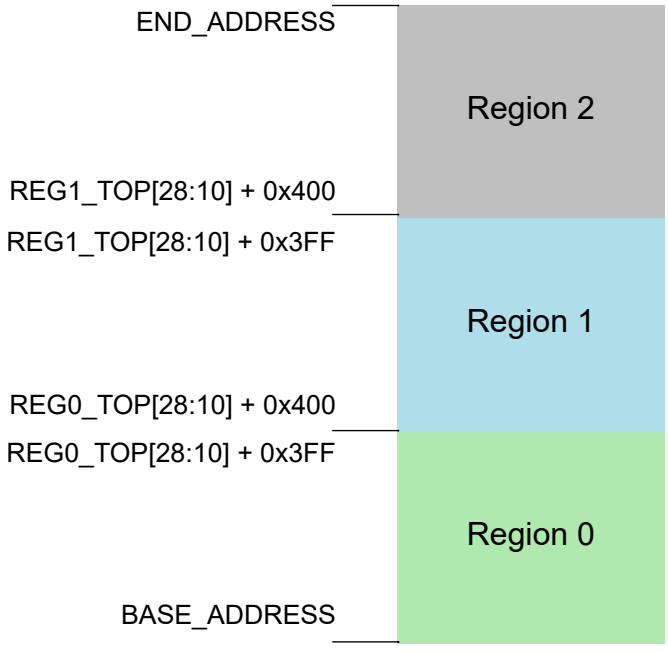


Figure 7. Cache memory regions

For CACHE64_POLSEL implementations that have multiple memory regions (more than one base address and end address), the regions are aliased across all of the memory regions. Therefore, the configured cache policies are also aliased across all of the memory regions.

7.3.2 Region priority

When you access an address, CACHE64_POLSEL attempts to match it to one of the regions in a sequential order that starts with region 0. In pseudocode, the region hit logic does this:

```
if (address < REG0_TOP + 0x3FF)
  address hits in Region 0;
else
  if (address < REG1_TOP + 0x3FF)
    address hits in Region 1;
  else
    address hits in Region 2;
```

NOTE

If you configure REG0_TOP and REG1_TOP to create overlapping regions, then the lowest number region in which the address hits, is used.

7.4 Application information

This section provides an example CACHE64_POLSEL configuration. The example assumes:

- The MCU has two memory regions mapped to CACHE64_POLSEL:
  - Nonsecure region: BASE_ADDRESS = 0800_0000h, END_ADDRESS = 0FFF_FFFFh
  - Secure region: BASE_ADDRESS = 1800_0000h, END_ADDRESS = 1FFF_FFFFh
- Region 0: 56 MB is configured as write-through.
- Region 1: 8 MB is configured as noncacheable.
- Region 2: The remaining memory region is unused and configured as invalid.

The following register settings are used to achieve this configuration:

- REG0_TOP = 037F_FC00h
- REG1_TOP = 03FF_FC00h
- POLSEL = 0000_0031h

The following figure shows the resulting memory configuration.

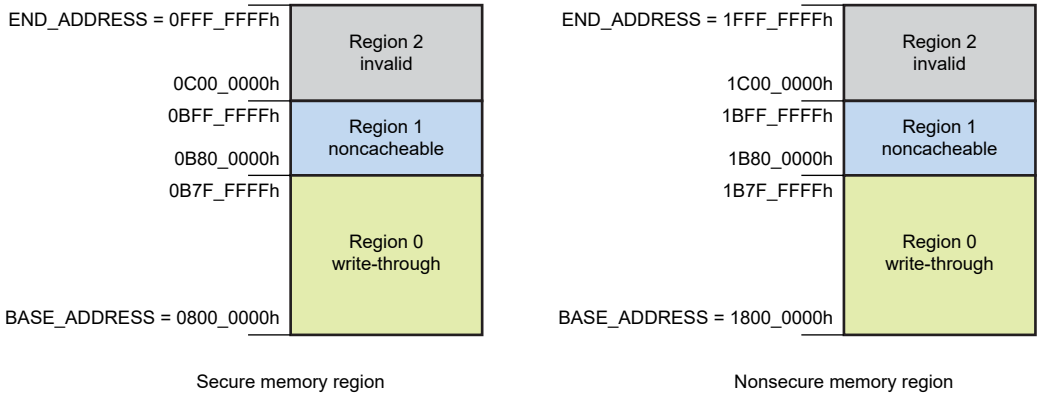


Figure 8. CACHE64_POLSEL configuration example

7.5 Memory map and register definition

The CACHE64_POLSEL registers are included in the register space of the CACHE64 module, utilizing addresses that CACHE64 does not use internally.

7.5.1 CACHE64_POLSEL register descriptions

7.5.1.1 CACHE64_POLSEL memory map

CACHE64_POLSEL0 base address: 4001_B000h

Offset	Register	Width (In bits)	Access	Reset value
14h	Region 0 Top Boundary (REG0_TOP)	32	RW	0AAA_A800h
18h	Region 1 Top Boundary (REG1_TOP)	32	RW	1555_5400h
1Ch	Policy Select (POLSEL)	32	RW	0000_0000h

7.5.1.2 Region 0 Top Boundary (REG0_TOP)

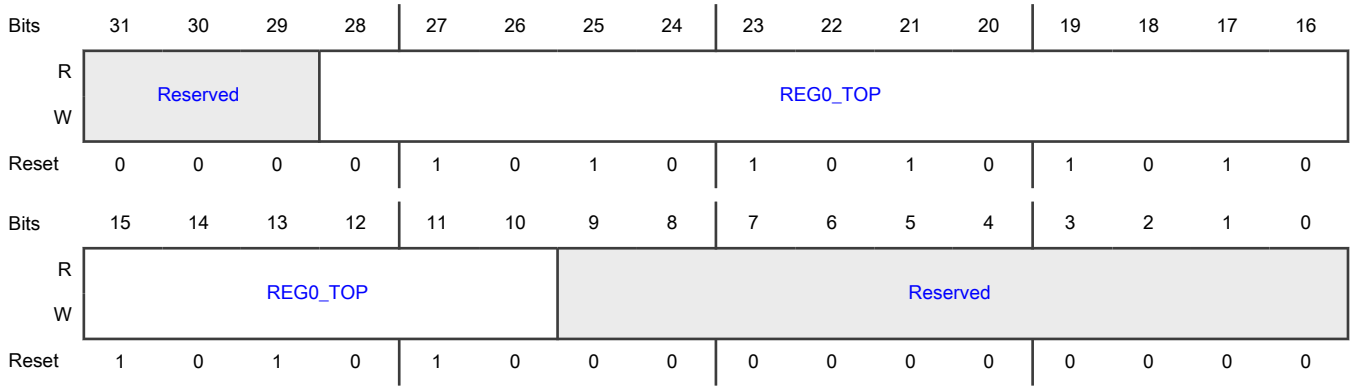
Offset

Register	Offset
REG0_TOP	14h

Function

Provides bits 28-10 of the region 0 compare address. The actual granularity implemented is 1 KB.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-10 REG0_TOP	Upper Limit Of Region 0 Provides bits 28-10 of the region 0 compare address. The value provided here is concatenated with 3FFh to create the full compare address. For example, if REG0_TOP[28:10] = 0b00000000000_111111, the region 0 end address is BASE_ADDRESS + 0000_FFFFh (64 KB).
9-0 —	Reserved

7.5.1.3 Region 1 Top Boundary (REG1_TOP)

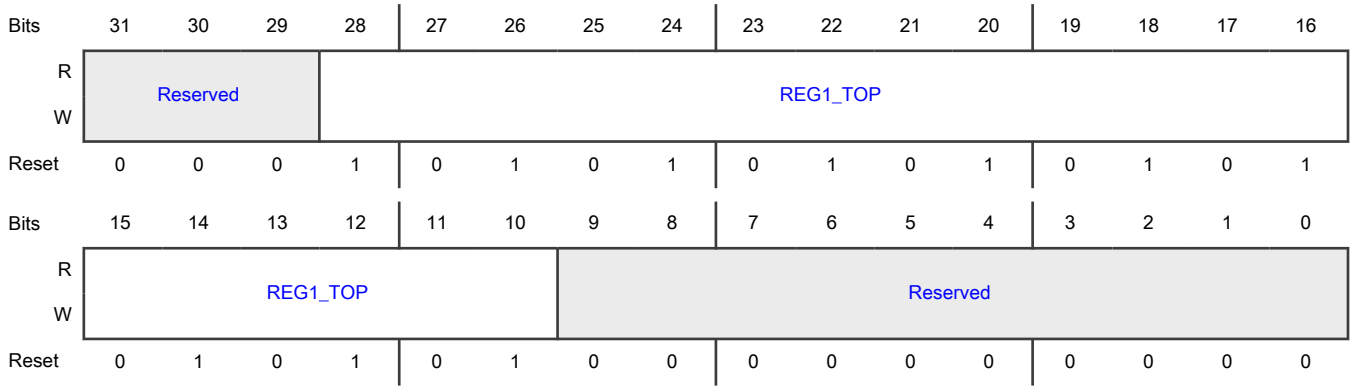
Offset

Register	Offset
REG1_TOP	18h

Function

Provides bits 28-10 of the region 1 compare address. The actual granularity implemented is 1 KB.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-10 REG1_TOP	Upper Limit Of Region 1 Provides bits 28-10 of the region 1 compare address. The value provided here is concatenated with 3FFh to create the full compare address. For example, if REG1_TOP[28:10] = 0b00000000000_111111, the region 0 end address is BASE_ADDRESS + 0000_FFFFh (64 KB).
9-0 —	Reserved

7.5.1.4 Policy Select (POLSEL)

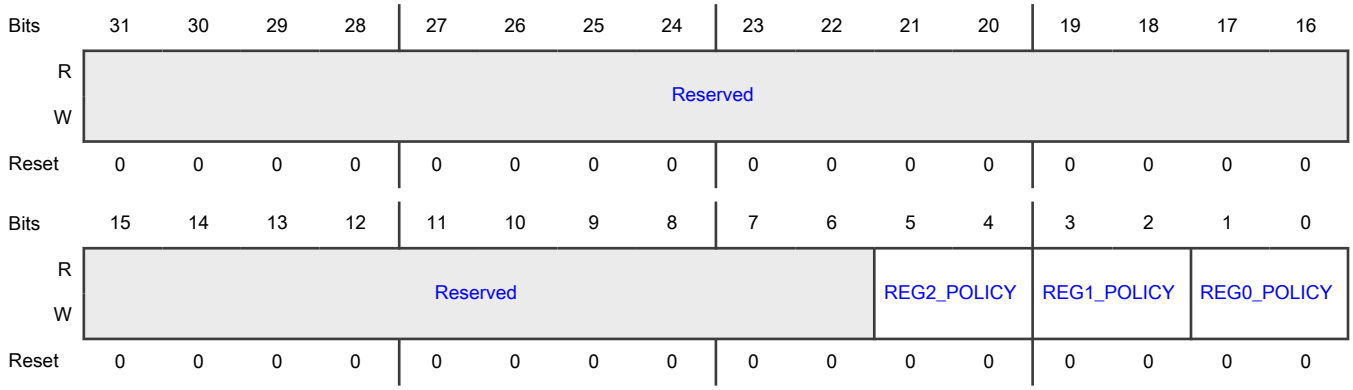
Offset

Register	Offset
POLSEL	1Ch

Function

Specifies the policy to be applied to all addresses falling within the designated space.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-4 REG2_POLICY	Policy Select For Region 2 Specifies the policy to be applied to all addresses falling within the space designated as region 2. 00b - Noncacheable 01b - Write-through 10b - Write-back 11b - Invalid
3-2 REG1_POLICY	Policy Select For Region 1 Specifies the policy to be applied to all addresses falling within the space designated as region 1. 00b - Noncacheable 01b - Write-through 10b - Write-back 11b - Invalid
1-0 REG0_POLICY	Policy Select For Region 0 Specifies the policy to be applied to all addresses falling within the space designated as region 0. 00b - Noncacheable 01b - Write-through 10b - Write-back 11b - Invalid

Chapter 8
Flash Memory Module (FMU)

8.1 Chip-specific FMU information

Table 36. Reference links to related information

Topic	Related module	Reference
Full description	FMU	Flash Memory Module
System memory map		Memory map
Clocking		Clock distribution

8.1.1 Module instances

This device contains one instance of the FMU module, FMU0.

8.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

8.1.3 Key terms

- Flash page — 128 bytes of flash main memory; represents the largest portion of the flash memory that can be programmed in one operation.
- Flash phrase — 16 bytes of flash main memory; represents the smallest portion of the flash memory that can be programmed in one operation.
- Flash sector — 8 KB flash main memory; represents the smallest portion of the flash memory that can be erased in one operation.

8.1.4 Internal flash

See Internal flash.

8.1.5 Salvage feature

The salvage feature is enabled in this MCU. Salvage involves using ECC to pass erase verify, allowing single-bit faults per phrase verified. See description of FSTAT[SALV_USED] for more details.

8.1.6 Register reset value

The reset value of the FSTAT and FCCOB0 registers might change depending on the device's BootROM settings.

8.1.7 FCTRL[RWSC] configuration

The configuration of Read Wait-State Control (RWSC) field of Flash Control Register (FCTRL) is shown in below table.

Table 37. FCTRL[RWSC] configuration in different run modes and speed

OD mode	SD Mode	MD Mode	RWSC
101-150 MHz	—	—	0011 (default value)
65-100 MHz, includes 100 MHz	65-100 MHz, includes 100 MHz	—	0010
37-64 MHz, includes 64 MHz	37-64 MHz, includes 64 MHz	25-50 MHz	0001
<= 36 MHz	<= 36 MHz	<= 24 MHz	0000

8.2 Overview

The flash module includes the following accessible memory regions:

- Program flash memory for vector space and code store
- Separate flash memory for parameter store

Flash memory is ideal for single-supply applications, permitting in-the-field erase and reprogramming operations without the need for any external high voltage power sources.

The flash module includes a command controller that executes commands to modify flash memory contents. An erased bit reads '1' and a programmed bit reads '0'. The programming operation is unidirectional; it can only move bits from the '1' state (erased) to the '0' state (programmed). Only the erase operation restores bits from '0' to '1'; bits cannot be programmed from a '0' to a '1'.

CAUTION

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash phrase or page is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

The standard shipping condition for flash memory is erased. Data loss over time may occur due to degradation of the erased ('1') states and/or programmed ('0') states. Therefore, it is recommended that flash memory be re-erased immediately prior to factory programming to ensure that the full data retention capability is achieved.

8.2.1 Block diagram

The block diagram of the flash module is shown in the following figure.

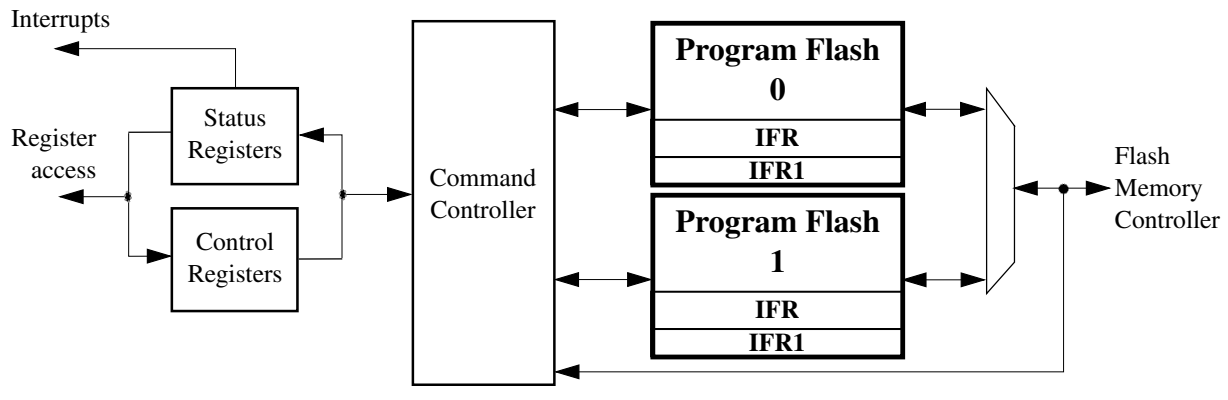


Figure 9. Flash block diagram

8.2.2 Features

The flash memory module provides the following features:

- Sector size of 8 Kbytes
- Single-bit error correction and double-bit error detection for each flash or IFR phrase
- Read-while-write flash access
- Command-based flash program, erase, and verify operations
- Internal high-voltage supply generator for flash program and erase operations
- Optional interrupt generation upon flash command completion
- Optional interrupt generation upon double-bit error detection during flash reads from the FMC

NOTE

See the chip configuration details for the exact amount of flash memory available.

8.3 Functional description

The following sections describe functional details of the flash module.

8.3.1 Operations

Operations are typically used to modify flash memory contents. The next sections describe:

- Command write sequence used to set flash command parameters and launch execution
- Available flash commands
- Commands impacted by Block Checker and FMC access protection
- Read-while-write capability

8.3.1.1 Command write sequence

Flash commands are accomplished using a command write sequence illustrated in Figure 10. The command controller monitors the transaction protection level on writes during the command write sequence, performs various checks on the command (FCCOB) content, and continues with command execution if all requirements are fulfilled.

Before launching a command, the ACCERR, PVIOL, and CMDABT flags in the FSTAT register must be zero and the CCIF flag must be one to verify that any previous command has completed. The FAIL flag in the FSTAT register is cleared by the command controller when a command is launched unless the FAIL flag was set during initialization due to FMU parameters not being loaded from IFR1. If CCIF is clear, the previous command execution is still active, a new command write sequence cannot be started, and all writes to the FCCOB registers are ignored.

8.3.1.1.1 Load the FCCOB Registers

The user must load the FCCOB registers with all parameters required by the specific flash command. The contents of any FCCOB register not used by the specific flash command are ignored except for FCCOB1 (command options) where undefined bits should be cleared. The individual registers that make up the FCCOB data set can be written in any order. All writes to the FCCOB registers in a command write sequence must occur at the same transaction protection level. Any write to the FCCOB registers that violates this rule will be ignored.

Addresses loaded into FCCOB registers assume the flash and IFR spaces start at 0000_0000h. For multi-block configurations, flash space is concatenated and IFR space is concatenated (see IFR description).

8.3.1.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing the FSTAT[CCIF] bit by writing a '1' to it. The write to clear CCIF must occur at the same transaction protection level as the writes used to load

the FCCOB registers. Any write to clear CCIF that violates this rule will be ignored. The CCIF flag remains clear until the flash command completes.

The FSTAT register contains a blocking mechanism which prevents a new flash command from launching (unable to clear CCIF) if the previous command resulted in the setting of an access error (ACCERR), protection violation (PVIOL), or active command abort flag (CMDABT). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write to clear these flags, the second write to clear CCIF. The write to clear these flags can occur at any transaction protection level.

8.3.1.1.3 Command Execution and Error Reporting

Flash command processing contains the following steps:

1. The command controller performs a series of parameter checks, if applicable, which are unique to each command. If the parameter check fails, the FSTAT[ACCERR] flag is set. ACCERR reports invalid instruction codes or options, out of bounds or misaligned addresses, or missing FMU parameters. In the case FMU parameters were not loaded from IFR1 during initialization, the ACCERR flag sets and the FSTAT[FAIL] flag remains set. Command processing never proceeds to execution when the parameter check fails. Instead, command processing is terminated after setting the FSTAT[CCIF] flag. Note that for program and erase operations, PEWEN will not set if FMU parameters were not loaded and PERDY will not set if program or erase related writes to flash or IFR space violate command requirements.
2. If necessary, the command controller performs a protection check to see if the command is allowed to execute on the requested memory space. If the protection check fails, the FSTAT[PVIOL] flag is set. Command processing never proceeds to execution when the protection check fails. Instead, command processing is terminated after setting the FSTAT[CCIF] flag.
3. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to verify, may occur during the execution phase and are reported in the FAIL flag. A flash command may have access errors, protection violations, and run-time errors, but the run-time errors are not seen until all access errors and protection violations have been corrected.
4. If FCTRL[ABTREQ] sets during command execution, the operation will abort with the FSTAT[CMDABT] flag set.
5. Command execution results, if applicable, are reported back to the user via the FCCOB registers.
6. The command controller sets FSTAT[CCIF] signifying that the command has completed.

CAUTION

If the FAIL flag is set along with ACCERR, FMU parameters were not loaded from IFR1 during initialization. In this case, it is recommended that the flash module be reinitialized. If the issue persists, flash commands will continue to set the ACCERR and FAIL flags.

The flow for a generic command write sequence is illustrated in the following figure.

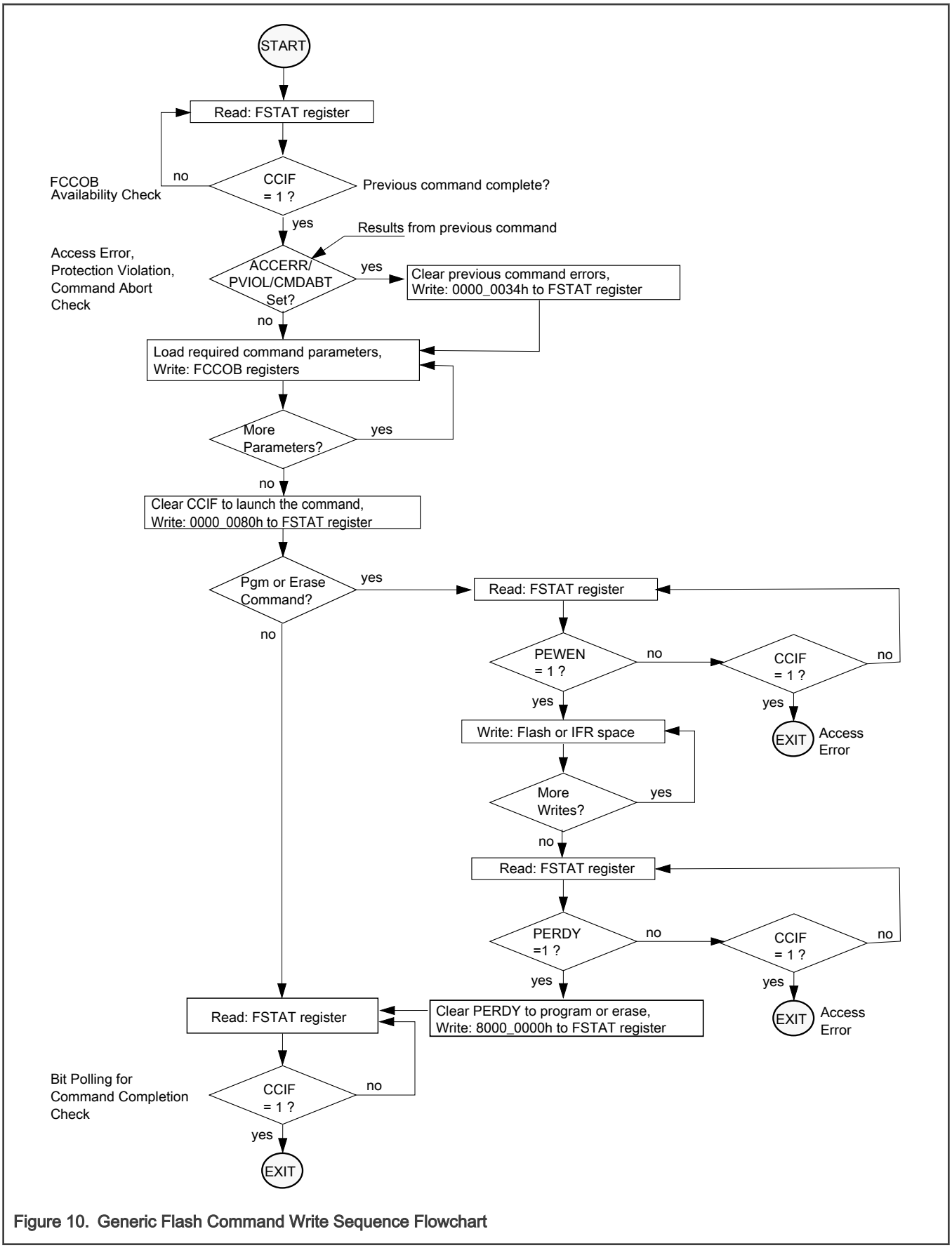


Figure 10. Generic Flash Command Write Sequence Flowchart

8.3.1.1.4 Aborting a Command Write Sequence or Command Operation

FSTAT[CMDP] sets at the start of a command write sequence, typically by writing to one of the FCCOB registers. With CMDP set, setting the FCTRL[ABTREQ] bit allows a user to abort a command write sequence and take control over a new command write sequence. The ABTREQ bit can only be set by a user with the same domain ID as indicated by FSTAT[CMDDID] and at the same or higher security/privilege protection level when compared to FSTAT[CMDPRT], the current owner of the command write sequence. While ABTREQ is set, CCIF cannot be cleared to launch a command.

If FSTAT[CCIF] is high when ABTREQ is set, the command write sequence will abort and ownership transferred to the user setting the ABTREQ bit. The FSTAT[CWSABT] flag will set and the CMDPRT field will be updated based on the write transaction that set the ABTREQ bit. The ABTREQ bit will then be cleared automatically.

If CCIF is low when ABTREQ is set, the command controller will abort execution of the current command operation, set FSTAT[CMDABT], and set CCIF. The CMDPRT field will be updated based on the write transaction that set the ABTREQ bit. The ABTREQ bit will then be cleared automatically. While CMDABT is set, CCIF cannot be cleared to launch a command. Aborting a program or erase operation may leave the flash or IFR space being modified in an indeterminate state.

NOTE

Aborting certain flash commands may take longer to abort than to execute. See the device data sheet for flash command timing specifications.

8.3.1.2 Flash commands

The following table summarizes the function of all flash commands. If any column is marked with an 'X', the flash command is relevant to that particular memory resource.

FCMD	Command	Program flash 0	IFR	Program flash 1[1]	Function
00h	Read 1s All (RD1ALL)	x	x	x	Verify that all flash and IFR space is erased
01h	Read 1s Block (RD1BLK)	x		x	Verify that a flash block is erased
02h	Read 1s Sector (RD1SCR)	x		x	Verify that a flash sector is erased
03h	Read 1s Page (RD1PG)	x		x	Verify that a flash page is erased
04h	Read 1s Phrase (RD1PHR)	x		x	Verify that a flash phrase is erased
05h	Read into MISR (RDMISR)	x		x	Generate MISR signature for range of flash pages
12h	Read 1s IFR Sector (RD1ISCR)		x		Verify that an IFR sector is erased
13h	Read 1s IFR Page (RD1IPG)		x		Verify that an IFR page is erased
14h	Read 1s IFR Phrase (RD1IPHR)		x		Verify that an IFR phrase is erased

Table continues on the next page...

[1] For module with dual flash blocks.

Table continued from the previous page...

FCMD	Command	Program flash 0	IFR	Program flash 1[1]	Function
15h	Read IFR into MISR (RDIMISR)		x		Generate MISR signature for range of IFR pages
23h	Program Page (PGMPG)	x	x	x	Program data to a flash or IFR page
24h	Program Phrase (PGMPHR)	x	x	x	Program data to a flash or IFR phrase
40h	Erase All (ERSALL)	x	x	x	Erase all flash and IFR space if enabled
42h	Erase Sector (ERSSCR)	x	x	x	Erase a flash sector

8.3.1.3 Flash commands impacted by Block Checker and FMC access protection

The following flash commands are impacted by Block Checker and FMC access protection:

- Program Phrase (PGMPHR)
- Program Page (PGMPG)
- Erase Sector (ERSSCR)

During execution of these flash commands, a sequence of AHB writes to flash or IFR space are required with validity checked by the Block Checker or FMC. When writes are enabled by the FMU, violations detected by the FMC are reported as an access error in the FMU. The other violations are handled by the Block Checker or FMC.

Table 38. Flash Write Access Checks

Flash Write Violation Condition	Block Checker	FMC	FMU
Flash or IFR address out-of-range	X		
Flash or IFR write permission check fails	X		
Write to IFR1 space		X	
Write width not 32-bit		X	
Writes not enabled by FSTAT[PEWEN]		X	
First write in sequence not properly aligned per FSTAT[PEWEN]			X
Non-sequential write after first aligned write in sequence			X
Bus permissions for the AHB write do not match the bus permissions of the APB write that launched the flash command			X

8.3.1.4 Read-while-write (RWW)

The following simultaneous accesses are allowed:

[1] For module with dual flash blocks.
[1] For module with dual flash blocks.

- The user may read from one program flash memory block while a command is active in another program flash memory block.

8.3.2 Flash command descriptions

This section describes all flash commands that can be launched by a valid command write sequence. The flash module sets the FSTAT[ACCERR] bit and aborts the command execution if any of the following illegal conditions occur:

- There is an unrecognized command code in the FCCOB FCMD field.
- There is an error in an FCCOB field for the specific command. Refer to the error handling table provided for each command.

The ACCERR, PVIOL, and CMDABT flags in the FSTAT register should be cleared prior to starting a command write sequence. As described in Launch the Command by Clearing CCIF, a new command cannot be launched while these flags are set.

Do not attempt to read a flash block while a flash command is running (CCIF = 0) on that same block. The flash module may return a transfer error to the FMC resulting in a bus fault. The exception to this rule is during program operations where the flash block is still readable after the command is launched up until the time the FSTAT[PERDY] flag is cleared.

CAUTION

Flash data must be in the erased state before being programmed. Cumulative programming of bits (adding more zeros) is not allowed.

8.3.2.1 Read 1s All Command

The Read 1s All command checks if all flash and IFR space are in the erased state.

Table 39. Read 1s All Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	00h (RD1ALL)

Upon clearing CCIF to launch the Read 1s All command, the command controller:

- verifies that all flash and IFR space are in the erased state

If all flash and IFR space are not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s All operation completes.

Table 40. Read 1s All Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.2 Read 1s Block Command

The Read 1s Block command checks if a flash block is in the erased state.

Table 41. Read 1s Block Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	01h (RD1BLK)
2	Block address

Upon clearing CCIF to launch the Read 1s Block command, the command controller:

- verifies that the selected flash block is in the erased state

If the selected flash block is not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s Block operation completes.

Table 42. Read 1s Block Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Address is not block aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.3 Read 1s Sector Command

The Read 1s Sector command checks if a flash sector is in the erased state.

Table 43. Read 1s Sector Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	02h (RD1SCR)
2	Sector address

Upon clearing CCIF to launch the Read 1s Sector command, the command controller:

- verifies that the selected flash sector is in the erased state

If the selected flash sector is not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s Sector operation completes.

Table 44. Read 1s Sector Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Address is not sector aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.4 Read 1s Page Command

The Read 1s Page command checks if a flash page is in the erased state.

Table 45. Read 1s Page Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	03h (RD1PG)
2	Page address

Upon clearing CCIF to launch the Read 1s Page command, the command controller:

- verifies that the selected flash page is in the erased state

If the selected flash page is not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s Page operation completes.

Table 46. Read 1s Page Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Address is not page aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.5 Read 1s Phrase Command

The Read 1s Phrase command checks if a flash phrase is in the erased state.

Table 47. Read 1s Phrase Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	04h (RD1PHR)
2	Phrase address

Upon clearing CCIF to launch the Read 1s Phrase command, the command controller:

- verifies that the selected flash phrase is in the erased state

If the selected flash phrase is not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s Phrase operation completes.

Table 48. Read 1s Phrase Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Address is not phrase aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.6 Read into MISR command

The Read into MISR operation generates a signature based on the contents of the selected flash memory using an embedded MISR.

Table 49. Read into MISR Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	05h (RDMISR) [7:0]
2	Starting flash page address
3	Ending flash phrase address
4	Word 0 of MISR seed
5	Word 1 of MISR seed
6	Word 2 of MISR seed
7	Word 3 of MISR seed
Returned values	
4	Word 0 of MISR signature
5	Word 1 of MISR signature
6	Word 2 of MISR signature
7	Word 3 of MISR signature

Upon clearing CCIF to launch the Read into MISR command, the command controller:

1. initializes the MISR with the seed provided
2. processes flash data starting with the starting flash page address provided
3. continues until the ending flash phrase address has been processed (must be the last phrase in a page)
4. returns the resulting signature into the FCCOB register bank unless an uncorrectable ECC fault is detected

The CCIF flag sets after the Read into MISR operation completes. If the operation is aborted by setting FCTRL[ABTREQ], a signature will not be returned if the abort occurs prior to the operation writing the signature.

MISR polynomial is $X^{128} + X^{126} + X^{101} + X^{99} + 1$.

Table 50. Read into MISR Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Starting address is not flash page aligned	FSTAT[ACCERR]
Ending address is not flash phrase aligned	FSTAT[ACCERR]
Ending address is not the last phrase in a page	FSTAT[ACCERR]
Starting address is larger than the ending address	FSTAT[ACCERR]
Requested range crosses a flash block boundary	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]

Table continues on the next page...

Table 50. Read into MISR Command Error Handling (continued)

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Uncorrectable ECC error is detected	FSTAT[FAIL]

8.3.2.7 Read 1s IFR Sector Command

The Read 1s IFR Sector command checks if an IFR sector is in the erased state.

Table 51. Read 1s IFR Sector Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	12h (RD1ISCR)
2	IFR sector address

Upon clearing CCIF to launch the Read 1s IFR Sector command, the command controller:

- verifies that the selected IFR sector is in the erased state

If the selected IFR sector is not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s IFR Sector operation completes.

Table 52. Read 1s IFR Sector Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Address is not IFR sector aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.8 Read 1s IFR Page Command

The Read 1s IFR Page command checks if an IFR page is in the erased state.

Table 53. Read 1s IFR Page Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	13h (RD1IPG)
2	IFR page address

Upon clearing CCIF to launch the Read 1s IFR Page command, the command controller:

- verifies that the selected IFR page is in the erased state

If the selected IFR page is not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s IFR Page operation completes.

Table 54. Read 1s IFR Page Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Address is not IFR page aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.9 Read 1s IFR Phrase Command

The Read 1s IFR Phrase command checks if an IFR phrase is in the erased state.

Table 55. Read 1s IFR Phrase Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	14h (RD1IPHR)
2	IFR phrase address

Upon clearing CCIF to launch the Read 1s IFR Phrase command, the command controller:

- verifies that the selected IFR phrase is in the erased state

If the selected IFR phrase is not in the erased state, the FSTAT[FAIL] flag is set. The CCIF flag sets after the Read 1s IFR Phrase operation completes.

Table 56. Read 1s IFR Phrase Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Address is not IFR phrase aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.10 Read IFR into MISR command

The Read IFR into MISR operation generates a signature based on the contents of the selected IFR space using an embedded MISR.

Table 57. Read IFR into MISR Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	15h (RDIMISR) [7:0]
2	Starting IFR page address

Table continues on the next page...

Table 57. Read IFR into MISR Command FCCOB Requirements (continued)

FCCOB Number	FCCOB Contents
3	Ending IFR phrase address
4	Word 0 of MISR seed
5	Word 1 of MISR seed
6	Word 2 of MISR seed
7	Word 3 of MISR seed
Returned values	
4	Word 0 of MISR signature
5	Word 1 of MISR signature
6	Word 2 of MISR signature
7	Word 3 of MISR signature

Upon clearing CCIF to launch the Read IFR into MISR command, the command controller:

1. initializes the MISR with the seed provided
2. processes IFR data starting with the starting IFR page address provided
3. continues until the ending IFR phrase has been processed (must be the last phrase in a page)
4. returns the resulting signature into the FCCOB register bank unless an uncorrectable ECC fault is detected

The CCIF flag sets after the Read IFR into MISR operation completes. If the operation is aborted by setting FCTRL[ABTREQ], a signature will not be returned if the abort occurs prior to the operation writing the signature.

MISR polynomial is $X^{128} + X^{126} + X^{101} + X^{99} + 1$.

Table 58. Read IFR into MISR Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Address is not valid	FSTAT[ACCERR]
Starting address is not IFR page aligned	FSTAT[ACCERR]
Ending address is not IFR phrase aligned	FSTAT[ACCERR]
Ending address is not the last phrase in a page	FSTAT[ACCERR]
Starting address is larger than the ending address	FSTAT[ACCERR]
Requested range crosses the IFR block boundary	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Uncorrectable ECC error is detected	FSTAT[FAIL]

8.3.2.11 Program Page command

The Program Page operation programs 32 words of data to a previously erased flash or IFR page.

CAUTION

The page must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a page is not allowed.

Table 59. Program Page Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	23h (PGMPG) [7:0]

Upon clearing CCIF to launch the Program Page command, the command controller:

1. sets FSTAT[PEWEN] to enable writes to the flash and IFR space in the system memory map
2. waits for the user to write 32 consecutive words to the flash or IFR space with the first write being page aligned
3. clears FSTAT[PEWEN] field
4. sets FSTAT[PERDY]
5. waits for the user to clear FSTAT[PERDY]
6. programs the data into the targeted page written to in step 2
7. verifies bits to be programmed are programmed to the program verify level

The CCIF flag sets after the Program Page operation completes. The Program Page operation can be aborted by setting FCTRL[ABTREQ]. If the operation is aborted, the operation terminates with the FSTAT[CMDABT] flag set. Once CCIF is set, the Program Page command may be repeated (same addresses, same data) to complete the program operation without erasing the sector containing the targeted page. Invalid writes to flash or IFR space based on Block Checker or FMC access protection are trapped outside of the flash module.

Table 60. Program Page Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Protection level of the writes to flash or IFR space do not match the protection level of the command write sequence	FSTAT[ACCERR]
Initial address is not page aligned and subsequent addresses phrase aligned	FSTAT[ACCERR]
Number of writes to flash or IFR space is more than expected	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.11.1 Page programming

The standard process of programming one or more flash or IFR pages is as follows:

1. If required, launch the Erase Sector command to erase the flash or IFR sector to be programmed (repeat for additional sectors).
2. Launch the Program Page command to enable writes to the flash space.
3. After the command controller sets FSTAT[PEWEN], write 32 consecutive words to the flash space with the first word page aligned. The data written to the flash space is not readable until the program operation has successfully completed.
4. After the flash page has been written, the command controller will stall and set FSTAT[PERDY].

5. Clear PERDY allowing the command controller to resume and program the 32 words into the flash or IFR space written to in step 3.
6. Wait for CCIF to set indicating the program operation has completed.
7. To program additional flash or IFR pages in previously erased flash memory, repeat steps 2 through 6.

8.3.2.12 Program Phrase command

The Program Phrase operation programs 4 words of data to a previously erased flash or IFR phrase.

CAUTION

The phrase must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a phrase is not allowed.

Table 61. Program Phrase Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	24h (PGMPHR) [7:0]

Upon clearing CCIF to launch the Program Phrase command, the command controller:

1. sets FSTAT[PEWEN] to enable writes to the flash and IFR space in the system memory map
2. waits for the user to write 4 consecutive words to the flash or IFR space with the first write being phrase aligned
3. clears FSTAT[PEWEN] field
4. sets FSTAT[PERDY]
5. waits for the user to clear FSTAT[PERDY]
6. programs the data into the targeted phrase written to in step 2.
7. verifies bits to be programmed are programmed to the program verify level

The CCIF flag sets after the Program Phrase operation completes. The Program Phrase operation can be aborted by setting FCTRL[ABTREQ]. If the operation is aborted, the operation terminates with the FSTAT[CMDABT] flag set. Once CCIF is set and the CMDABT flag is cleared, the Program Phrase command may be repeated (same addresses, same data) to complete the program operation without erasing the sector containing the targeted phrase. Invalid writes to flash or IFR space based on Block Checker or FMC access protection are trapped outside of the flash module.

Table 62. Program Phrase Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Protection level of the writes to flash or IFR space do not match the protection level of the command write sequence	FSTAT[ACCERR]
Address is not phrase aligned	FSTAT[ACCERR]
Number of writes to flash or IFR space is more than expected	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.12.1 Phrase programming

The standard process of programming one or more flash or IFR phrases is as follows:

1. If required, launch the Erase Sector command to erase the flash or IFR sector to be programmed (repeat for additional sectors).
2. Launch the Program Phrase command to enable writes to the flash and IFR space.
3. After the command controller sets FSTAT[PEWEN], write 4 consecutive words to the flash or IFR space with the first word phrase aligned. The data written to the flash or IFR space is not readable until the program operation has successfully completed.
4. After the flash or IFR phrase has been written, the command controller will stall the operation and set FSTAT[PERDY].
5. Clear PERDY allowing the command controller to resume and program the 4 words into the flash or IFR space written to in step 3.
6. Wait for CCIF to set indicating the program operation has completed.
7. To program additional flash or IFR phrases in previously erased flash memory, repeat steps 2 through 6.

8.3.2.13 Erase All command

The Erase All operation erases all flash and IFR space if enabled by the MCU. If salvage is enabled, the erase operation will not fail due to detected single-bit ECC faults.

Table 63. Erase All Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	40h (ERSALL) [7:0]

Upon clearing CCIF to launch the Erase All command, the command controller:

1. erases flash block 1 sector-by-sector
2. verifies that flash block 1 is erased; if not, skips IFR space in block 1 and all of block 0
3. erases IFR space in block 1 sector-by-sector
4. verifies IFR space in block 1 is erased; if not, skips all of block 0
5. erases flash block 0 sector-by-sector
6. verifies flash block 0 is erased; if not, skips IFR space in block 0
7. erases IFR space in block 0 sector-by-sector
8. verifies IFR space in block 0 is erased
9. clears FSTAT[CMDP] flag

The CCIF flag sets after the Erase All operation completes. If either the flash or IFR space is not in the erased state, FSTAT[FAIL] sets.

Table 64. Erase All Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Command is disabled by the MCU	FSTAT[PVIOL]
Operation is aborted	FSTAT[CMDABT]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.2.14 Erase Sector command

The Erase Sector operation erases the IFR sector to prepare the sector for programming.

Table 65. Erase Sector Command FCCOB Requirements

FCCOB Number	FCCOB Contents
0	42h (ERSSCR) [7:0]

Upon clearing CCIF to launch the Erase Sector command, the command controller:

1. sets FSTAT[PEWEN] to enable writes to the flash and IFR space in the system memory map
2. waits for the user to write 4 consecutive words to the flash or IFR space with the first write being phrase (or sector) aligned
3. clears FSTAT[PEWEN] field
4. sets FSTAT[PERDY]
5. waits for the user to clear FSTAT[PERDY]
6. erases the selected flash or IFR sector
7. verifies the selected flash or IFR sector is erased

If the selected flash or IFR sector is not in the erased state, FSTAT[FAIL] is set. The CCIF flag sets after the Erase Sector operation completes. The Erase Sector operation can be aborted by setting FCTRL[ABTREQ]. If the operation is aborted, the operation terminates with the FSTAT[CMDABT] flag set. Once CCIF is set, the Erase Sector command may be repeated to complete the erase operation. Invalid writes to flash or IFR space based on Block Checker or FMC access protection are trapped outside of the flash module.

Table 66. Erase Sector Command Error Handling

Error Condition	Error Bit
FMU parameters not loaded from IFR1 during initialization	FSTAT[ACCERR]
Protection level of the writes to flash or IFR space do not match the protection level of the command write sequence	FSTAT[ACCERR]
Address is not flash or IFR phrase aligned	FSTAT[ACCERR]
Operation is aborted	FSTAT[CMDABT]
IFR Sector is protected (see FCNFG register)	FSTAT[PVIOL]
FMU parameters not loaded from IFR1 during initialization	FSTAT[FAIL]
Verify fails	FSTAT[FAIL]

8.3.3 Mass Erase

If enabled by the MCU, MCU resources can be used to launch a mass erase operation without accessing the flash registers. If salvage is enabled, the erase operation will not fail due to detected single-bit ECC faults.

To know how to enable/disable mass erase for your device, refer the chip-specific section.

The status of the mass erase request is reflected in the FCNFG[ERSREQ] bit. When launched, the mass erase operation occurs regardless of the state of the CMDP, ACCERR, PVIOL, or CMDABT flags in the FSTAT register but requires FSTAT[CCIF] to be set and the MCU to enable mass erase. When mass erase is launched with CCIF set, the CMDP, CCIF, CWSABT, PVIOL, and CMDABT flags in the FSTAT register clear. The ACCERR and FAIL flags also clear unless the FMU parameters were not loaded from IFR1 during initialization in which case the mass erase operation terminates without performing any erase of flash memory.

Upon successful launch of the mass erase operation, the command controller (same steps as the Erase All command):

1. erases flash block 1 sector-by-sector
2. verifies that flash block 1 is erased; if not, skips IFR space in block 1 and all of block 0
3. erases IFR space in block 1 sector-by-sector
4. verifies IFR space in block 1 is erased; if not, skips all of block 0
5. erases flash block 0 sector-by-sector
6. verifies flash block 0 is erased; if not, skips IFR space in block 0
7. erases IFR space in block 0 sector-by-sector
8. verifies IFR space in block 0 is erased

The CCIF flag sets and the ERSREQ bit clears once the operation completes and the FSTAT[FAIL] flag sets if any verify step fails during the mass erase operation. This method of erasing the flash memory cannot be aborted by setting FCTRL[ABTREQ].

8.3.4 Modes

8.3.4.1 Sleep mode

If the CPU enters sleep mode while a flash command is executing, the flash module can wake the CPU via the command complete interrupt (see Interrupts).

8.3.4.2 Non-Active Power mode

If a flash command is active (CCIF = 0) when the MCU requests a non-active power mode, the command execution completes before the MCU is allowed to enter the non-active power mode.

8.3.5 Clocking

This module has no clocking considerations.

8.3.6 Interrupts

The flash module can generate interrupt requests to the MCU upon the occurrence of various flash module events. These interrupt events and their associated status and control bits are shown in the following table.

Table 67. Flash Module Interrupt Sources

Flash Event	Readable Status Bit	Interrupt Enable Bit
Flash Command Complete	FSTAT[CCIF]	FCNFG[CCIE]
Flash ECC Fault Detected	FSTAT[DFDIF]	FCNFG[DFDIE]

NOTE

Vector addresses and their relative interrupt priority are determined at the MCU level.

8.4 External signals

The flash module contains no signals that connect off-chip.

8.5 Initialization

On each reset, the command controller initializes the following based on IFR1 content:

1. FCTRL[RWSC]
2. FMU Parameters (FMUPARM)

The FSTAT[CCIF] flag is cleared throughout the initialization flow. The flash module also holds off all access to flash registers and flash memory during initialization. Completion of the initialization flow is marked by setting FSTAT[CCIF]. If an uncorrectable ECC error is detected during initialization, the FSTAT[FAIL] flag sets. The FAIL flag also sets if FMU parameters are not loaded from IFR1 during initialization. In this case, attempts to launch a flash command terminate with the ACCERR and FAIL flags set. The recommendation is to refrain from launching flash commands if the FAIL flag sets after initialization.

If a reset occurs while any flash command is in progress, that command is immediately aborted. The state of the flash memory or IFR space being programmed or erased is not guaranteed. Command operations do not automatically resume after exiting reset.

8.6 Memory map and registers

This section describes the flash memory map for IFR/IFR1 space and registers. See the MCU system memory map for the location of all flash memory space and registers. Data read from unimplemented memory space in the flash module is undefined.

8.6.1 IFR description

Each flash block contains IFR space separate from the main flash memory. IFR space is memory mapped (see the MCU system memory map). IFR space is divided into IFR pages programmable using the Program Page command. IFR pages are further divided into IFR phrases programmable using the Program Phrase command. Once programmed, IFR phrases can be reprogrammed after a successful erase using the Erase Sector command operation unless the IFR sector is protected by the FCNFG register.

Table 68. IFR map

Offset Byte Address	IFR Field Size (Bytes)
Block 0 IFR Sector 0	
0000h-1FFFh	8,192
Block 0 IFR Sector 1	
2000h-3FFFh	8,192
Block 0 IFR Sector 2	
4000h-5FFFh	8,192
Block 0 IFR Sector 3	
6000h-7FFFh	8,192
Block 1 IFR Sector 0	
8000h-9FFFh	8,192
Block 1 IFR Sector 1	
A000h-BFFFh	8,192
Block 1 IFR Sector 2	
C000h-DFFFh	8,192
Block 1 IFR Sector 3	
E000h-FFFFh	8,192

Each flash block also contains IFR1 space separate from the main flash memory. IFR1 space in program flash block 0 is memory mapped for read-only access (see the MCU system memory map). IFR1 space cannot be erased or programmed by the user.

Table 69. IFR1 map

Offset Byte Address	IFR1 Field Size (Bytes)
Block 0 IFR1 Sector	
0000h-15FFh	5,632
1600h-16FFh	256
1700h-1FFFh	2,304
Block 1 IFR1 Sector	
2000h-3FFFh	8,192

8.6.2 Register descriptions

The flash module contains a set of memory-mapped control and status registers.

NOTE

While a command is running (FSTAT[CCIF]=0), register writes are allowed to FSTAT[PERDY,DFDIF,CWSABT], FCNFG, and FCTRL[ABTREQ,FDFD,RWSC].

8.6.2.1 Flash register descriptions

8.6.2.1.1 Flash memory map

FMU0.MSF1 base address: 4004_3000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Flash Status Register (FSTAT)	32	RW	See section
4h	Flash Configuration Register (FCNFG)	32	RW	See section
8h	Flash Control Register (FCTRL)	32	RW	See section
10h	Flash Common Command Object Registers (FCCOB0)	32	RW	0000_0000h
14h	Flash Common Command Object Registers (FCCOB1)	32	RW	0000_0000h
18h	Flash Common Command Object Registers (FCCOB2)	32	RW	0000_0000h
1Ch	Flash Common Command Object Registers (FCCOB3)	32	RW	0000_0000h
20h	Flash Common Command Object Registers (FCCOB4)	32	RW	0000_0000h
24h	Flash Common Command Object Registers (FCCOB5)	32	RW	0000_0000h
28h	Flash Common Command Object Registers (FCCOB6)	32	RW	0000_0000h
2Ch	Flash Common Command Object Registers (FCCOB7)	32	RW	0000_0000h

8.6.2.1.2 Flash Status Register (FSTAT)

Offset

Register	Offset
FSTAT	0h

Function

The FSTAT register reports the operational status of the flash module.

NOTE

When set, ACCERR, PVIOL, and CMDABT flags prevent the launch of any more commands until the flag is cleared (by writing a one to it).

CAUTION

When clearing the DFDIF flag, be careful not to use read-modify-write on the entire FSTAT register and inadvertently clear CCIF to launch a command.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PERDY	0					PEWEN		0						SALV_US...	DFDIF
W	W1C															W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDDID				CMDP	0	CMDPRT		CCIF	CWSABT	ACCERR	PVIOL	0	CMDABT	0	FAIL
W									W1C	W1C	W1C	W1C		W1C		
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	u

Fields

Field	Function
31 PERDY	Program-Erase Ready Control/Status Flag The PERDY flag is set by the command controller when a program or sector erase command operation has successfully completed the write phase. The command controller will stall the command operation until the user clears the PERDY flag by writing a 1 to PERDY. Failure to clear PERDY will leave the command controller stalled but the flash or IFR space targeted by the command operation will remain readable by the FMC. If FCTRL[ABTREQ] is set while the PERDY flag is set, the command controller will clear PERDY, abort the command operation, and set both CCIF and CMDABT. 0b - Program or sector erase command operation not stalled 1b - Program or sector erase command operation ready to execute

Table continues on the next page...

Table continued from the previous page...

Field	Function
30-26 —	Reserved
25-24 PEWEN	Program-Erase Write Enable Control During program or sector erase command operations, writes are enabled to flash or IFR space in the system memory map. For phrase programming or sector erase, writes are enabled for 4 consecutive words (16 bytes) with address phrase aligned. For page programming, writes are enabled for 8 consecutive phrases (128 bytes) with address page aligned. For programming, data written to erased flash or IFR space for program commands is not available until the program operation has successfully completed. PEWEN will not assert if FMU parameters were not loaded from IFR1 during initialization. 00b - Writes are not enabled 01b - Writes are enabled for one flash or IFR phrase (phrase programming, sector erase) 10b - Writes are enabled for one flash or IFR page (page programming) 11b - Reserved
23-18 —	Reserved
17 SALV_USED	Salvage Used for Erase operation This flag indicates salvage was used during the last Erase Sector, Erase All, or Mass Erase operation and is valid at the end of the operation. Salvage involves using ECC to pass erase verify allowing single-bit faults per phrase verified. This flag remains valid until any flash command or Mass Erase operation launches at which time the flag clears. Use of salvage does not guarantee that the erase operation succeeds but only that salvage was used on at least one phrase. 0b - Salvage not used during last operation 1b - Salvage used during the last erase operation
16 DFDIF	Double Bit Fault Detect Interrupt Flag The DFDIF flag indicates an uncorrectable ECC fault was detected during a valid flash read access from the FMC. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect. 0b - Double bit fault not detected during a valid flash read access 1b - Double bit fault detected (or FCTRL[FDFD] is set) during a valid flash read access
15-12 CMDDID	Command domain ID While the CMDP flag is set, the CMDDID bits reflect the domain ID of the write transaction controlling a command write sequence or aborting a command operation. Writes to load an FCCOB register, clear CCIF, or clear PERDY must have the same domain ID as the write transaction that started the command write sequence. The CMDDID bits persist after the CMDP flag clears until the CMDP flag sets again at the start of a new command write sequence. Writes to clear the CWSABT, ACCERR, PVIOL, or CMDABT flags are not influenced by the state of the CMDDID bits.
11	Command protection status flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
CMDP	The CMDP flag sets when a command write sequence starts with a write to load an existing FCCOB register or clear CCIF while ACCERR, PVIOL, and CMDABT are clear. Clearing the CWSABT, ACCERR, PVIOL, or CMDABT flags as part of a command write sequence will not set the CMDP flag. While the CMDP flag is set, the protection level of any subsequent write transaction to load an FCCOB register, clear CCIF, or clear PERDY must match the CMDPRT bits and the domain ID must match the CMDDID bits. The CMDP flag remains set until cleared when CCIF is set after a command operation or Power Down recovery completes unless the CMDABT flag is set. The CMDP flag is cleared even if the operation ends with ACCERR or PVIOL set. When the CMDP flag is cleared, the CMDPRT and CMDDID bits remain unchanged. The CMDP flag is also cleared by a mass erase request with CCIF set. 0b - Command protection level and domain ID are stale 1b - Command protection level (CMDPRT) and domain ID (CMDDID) are set
10 —	Reserved
9-8 CMDPRT	Command protection level While the CMDP flag is set, the CMDPRT bits reflect the secure/privilege protection level of the write transaction controlling a command write sequence or aborting a command operation. Writes to load an existing FCCOB register, clear CCIF, or clear PERDY must occur with the same protection level as indicated by the CMDPRT bits. Writes to clear the CWSABT, ACCERR, PVIOL, or CMDABT flags are not influenced by the state of the CMDPRT bits. If FCTRL[ABTREQ] is set while CCIF is high or low, the CMDPRT bits will change to reflect the security/ privilege protection level of the write transaction that set the ABTREQ bit. 00b - Secure, normal access 01b - Secure, privileged access 10b - Nonsecure, normal access 11b - Nonsecure, privileged access
7 CCIF	Command Complete Interrupt Flag The CCIF flag indicates that a flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a flash command. The CCIF flag stays low until the command completes. The CCIF flag is cleared by a mass erase request with CCIF set. The CCIF flag is also cleared upon entry into Deep Sleep, Power Down, and Deep Power Down modes and stays low until the end of recovery from these non-active power modes. The CCIF flag is reset to 0 but is set to 1 at the end of flash initialization. 0b - Flash command, initialization, or power mode recovery in progress 1b - Flash command, initialization, or power mode recovery has completed
6 CWSABT	Command Write Sequence Abort Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The CWSABT flag indicates whether a request to abort a command write sequence prior to command launch has been granted. If FCTRL[ABTREQ] is set while CCIF is high, the CWSABT flag will get set. Once CWSABT is set, ABTREQ will get cleared. While the CWSABT flag is set, ABTREQ cannot be set. The CWSABT flag is cleared by writing a 1 to CWSABT while CCIF is set or clear and ABTREQ is clear. Writing a 0 to the CWSABT flag has no effect. The CWSABT flag is cleared by a mass erase request with CCIF set. 0b - Command write sequence not aborted 1b - Command write sequence aborted
5 ACCERR	Command Access Error Flag The ACCERR flag indicates an illegal attempt was made to launch a flash command due to a violation of the command write sequence or by providing invalid command parameters. The ACCERR flag also sets during command execution if FMU parameters were not loaded from IFR1 during initialization. While the ACCERR flag is set, the CCIF flag cannot be cleared to launch a command. The ACCERR flag is cleared by writing a 1 to ACCERR while CCIF is set. Writing a 0 to the ACCERR flag has no effect. The ACCERR flag is cleared by a mass erase request with CCIF set but is not cleared during Power Down recovery. 0b - No access error detected 1b - Access error detected
4 PVIOL	Command Protection Violation Flag The PVIOL flag indicates an attempt was made to modify a protected area of flash memory during a command operation. While the PVIOL flag is set, the CCIF flag cannot be cleared to launch a command. The PVIOL flag is cleared by writing a 1 to PVIOL while CCIF is set. Writing a 0 to the PVIOL flag has no effect. The PVIOL flag is cleared by a mass erase request with CCIF set but is not cleared during Power Down recovery. 0b - No protection violation detected 1b - Protection violation detected
3 —	Reserved
2 CMDABT	Command Abort Flag The CMDABT flag indicates FCTRL[ABTREQ] was set during a command operation while CCIF was clear. This event will result in the termination of the operation unless it occurs during the exit routine of the operation. Once CMDABT is set, ABTREQ will get cleared. While the CMDABT flag is set, ABTREQ cannot be set and the CCIF flag cannot be cleared to launch a command. The CMDABT flag is cleared by writing a 1 to CMDABT while CCIF is set and ABTREQ is clear. Writing a 0 to the CMDABT flag has no effect. The CMDABT flag is cleared by a mass erase request with CCIF set.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No command abort detected 1b - Command abort detected
1 —	Reserved
0 FAIL	Command Fail Flag The FAIL flag is set if an error is detected during execution of a flash command, mass erase operation, or flash initialization. If the FAIL flag sets along with ACCERR after launching a flash command or mass erase operation, FMU parameters were not loaded from IFR1 during initialization and a clean POR where FMU parameters are successfully loaded is required to clear the FAIL flag. As a status flag, this bit cannot (and need not) be cleared by the user like some of the other flags in this register. The value of the FAIL flag for "command-N" is valid only at the end of the "command-N" execution when CCIF=1 and before the next command has been launched. At some point during the execution of "command-N+1," the FAIL flag is cleared unless FMU parameters were not loaded from IFR1 during initialization. The FAIL flag is cleared by a mass erase request with CCIF set unless FMU parameters were not loaded from IFR1 during initialization. The FAIL flag is not cleared during Power Down recovery. 0b - Error not detected 1b - Error detected

8.6.2.1.3 Flash Configuration Register (FCNFG)

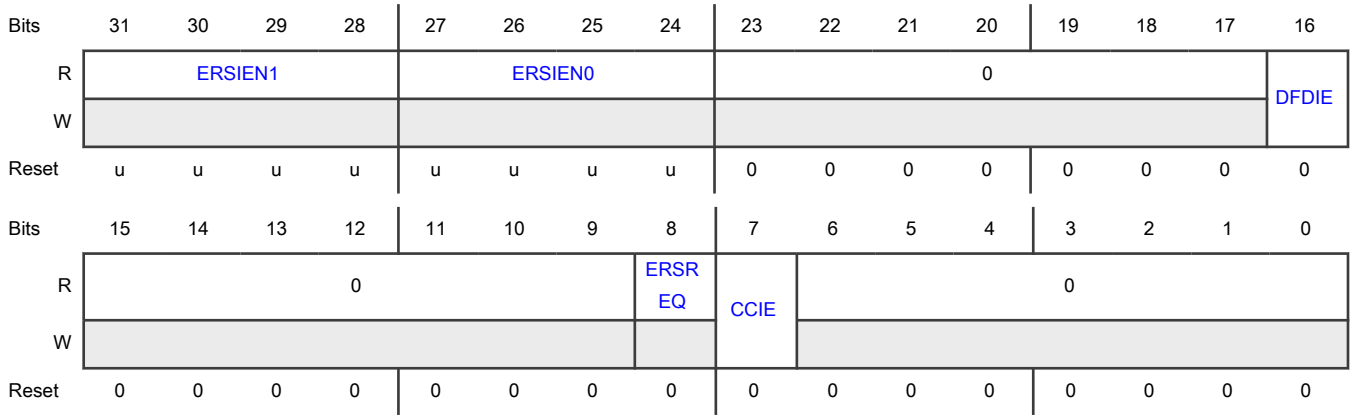
Offset

Register	Offset
FCNFG	4h

Function

This register provides information on the current functional state of the flash module.

Diagram



Fields

Field	Function
31-28 ERSIEN1	Erase IFR Sector Enable - Block 1 (for dual block configs) This field controls the ability to erase the IFR sectors in block 1 using the ERSSCR command. These bits are loaded from sideband signals during initialization. Note: These bits have no effect on single block configurations. Note: The ERSALL command and MCU mass erase request do not adhere to these restrictions. X equals 0, 1, 2, or 3. Any and all bits in the field are independently configured. 0000b - Block 1 IFR Sector X is protected from erase by ERSSCR command 0001b - Block 1 IFR Sector X is not protected from erase by ERSSCR command
27-24 ERSIEN0	Erase IFR Sector Enable - Block 0 This field controls the ability to erase the IFR sectors in block 0 using the ERSSCR command. These bits are loaded from sideband signals during initialization. Note: The ERSALL command and MCU mass erase request do not adhere to these restrictions. X equals 0, 1, 2, or 3. Any and all bits in the field are independently configured. 0000b - Block 0 IFR Sector X is protected from erase by ERSSCR command 0001b - Block 0 IFR Sector X is not protected from erase by ERSSCR command
23-17 —	Reserved
16 DFDIE	Double Bit Fault Detect Interrupt Enable The DFDIE bit controls interrupt generation when an uncorrectable ECC fault is detected during a valid flash read access from the platform flash controller. 0b - Double bit fault detect interrupt disabled 1b - Double bit fault detect interrupt enabled. An interrupt request is generated whenever the FSTAT[DFDIF] flag is set.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-9 —	Reserved
8 ERSREQ	Mass Erase Request This bit indicates whether a sideband request has been received by the command controller to execute the Mass Erase operation. ERSREQ is not directly writable but sets when a Mass Erase request is received by the flash module while CCIF is set. ERSREQ is cleared by the command controller when the operation completes. 0b - No request or request complete 1b - Request to run the Mass Erase operation
7 CCIE	Command Complete Interrupt Enable The CCIE bit controls interrupt generation when a flash command completes. 0b - Command complete interrupt disabled 1b - Command complete interrupt enabled. An interrupt request is generated whenever the FSTAT[CCIF] flag is set.
6-0 —	Reserved

8.6.2.1.4 Flash Control Register (FCTRL)

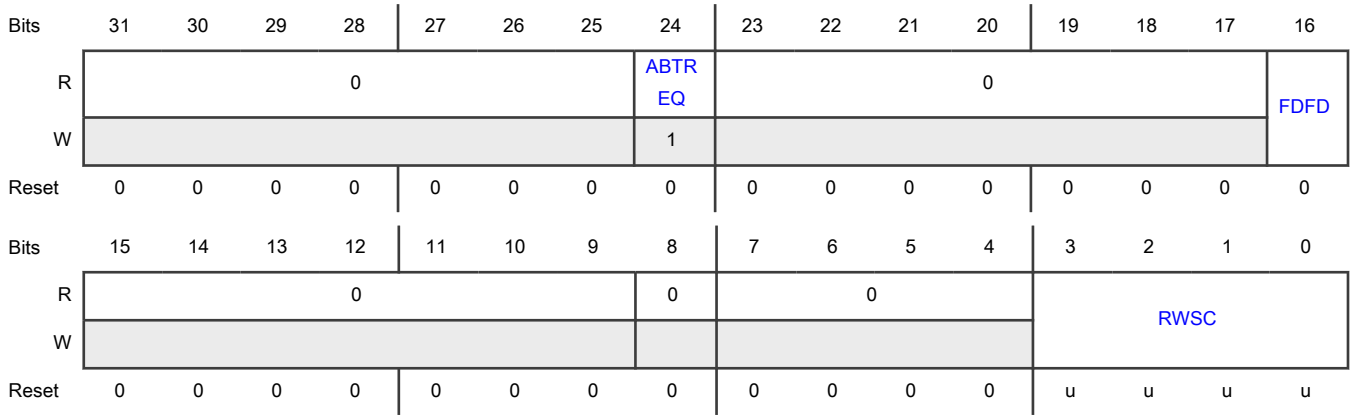
Offset

Register	Offset
FCTRL	8h

Function

The flash control register controls activity associated with flash memory reads and command operations.

Diagram



Fields

Field	Function
31-25 —	Reserved
24 ABTREQ	Abort Request With FSTAT[CMDP] set, setting the ABTREQ bit allows a user to abort a command write sequence or command operation. The ABTREQ bit can only be set by a user with the same domain ID as indicated by FSTAT[CMDDID] and the same or higher security/privilege protection level than is indicated by FSTAT[CMDPRT]. While ABTREQ is set, CCIF cannot be cleared to launch a command. If FSTAT[CCIF] is high when ABTREQ is set, the command write sequence will abort and ownership transferred to the user setting the ABTREQ bit. The FSTAT[CWSABT] flag will set and the CMDPRT field will be updated based on the write transaction that sets the ABTREQ bit. The ABTREQ bit will then be cleared automatically. If FSTAT[CCIF] is low when ABTREQ is set, the command operation will abort and ownership transferred to the user setting the ABTREQ bit. The FSTAT[CMDABT] flag will set and the CMDPRT field will be updated based on the write transaction that sets the ABTREQ bit. The ABTREQ bit will then be cleared automatically. While CWSABT or CMDABT are high, writes to ABTREQ will be ignored. 0b - No request to abort a command write sequence 1b - Request to abort a command write sequence
23-17 —	Reserved
16 FDFD	Force Double Bit Fault Detect The FDFD bit enables the user to emulate the setting of the FSTAT[DFDIF] flag to check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. 0b - FSTAT[DFDIF] sets only if a double bit fault is detected during a valid flash read access from the platform flash controller

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - FSTAT[DFDIF] sets during any valid flash read access from the platform flash controller. An interrupt request is generated if the DFDIE bit is set.
15-9 —	Reserved
8 —	Reserved
7-4 —	Reserved
3-0 RWSC	Read Wait-State Control These bits control the number of wait-states added to account for the ratio of system clock period to flash access time during full speed power mode. Ratios greater than one require non-zero settings for the RWSC field for proper flash accesses. The required settings are documented in the device data sheet. 0h - no additional wait-states are added (single cycle access) 1h - 1 additional wait-state is added 2h - 2 additional wait-states are added ... Fh - 15 additional wait-states are added These bits are loaded from IFR1 during initialization. **NOTE** The default FCTRL[RWSC] is set to a value appropriate for the chip's max operating frequency. Refer to the chip-specific section for the default value.

8.6.2.1.5 Flash Common Command Object Registers (FCCOB0 - FCCOB7)

Offset

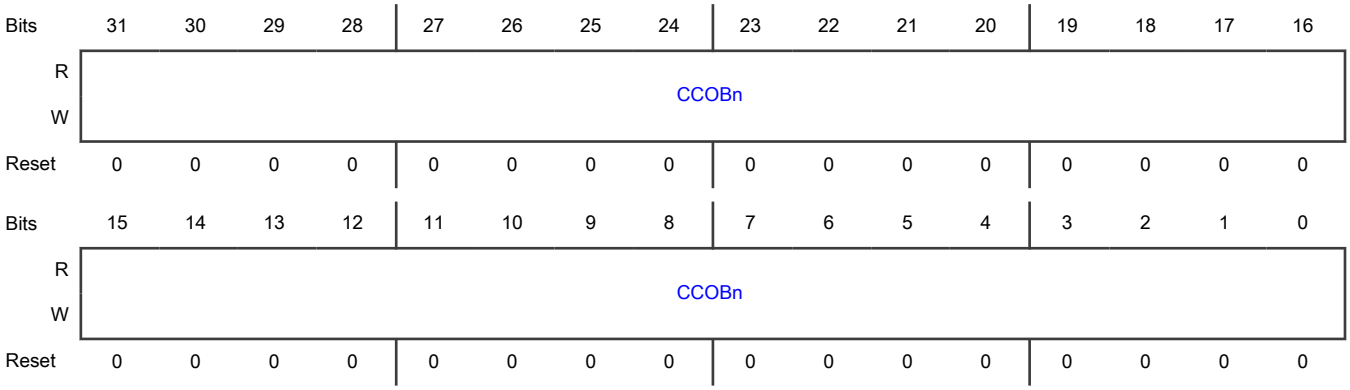
For a = 0 to 7:

Register	Offset
FCCOBa	10h + (a × 4h)

Function

The FCCOB register group provides fields for command codes and parameters. The individual words within the set append a 0-7 hex identifier to the FCCOB register name: FCCOB0, FCCOB1, ..., FCCOB7.

Diagram



Fields

Field	Function
31-0 CCOBn	CCOBn The FCCOB register group provides a command code and relevant parameters to the command controller. The individual registers that compose the FCCOB data set can be written in any order, but you must provide all needed values, which vary from command to command. First, set up all required FCCOB fields and then initiate the command execution by writing a 1 to the FSTAT[CCIF] bit. This clears the CCIF bit, which locks all FCCOB parameter fields and they cannot be changed by the user until the command completes (CCIF returns to 1). No command buffering or queueing is provided; the next command can be loaded only after the current command completes. Some commands return information to the FCCOB registers. Any values returned to FCCOB are available for reading after the FSTAT[CCIF] flag returns to 1. The following table shows a generic flash command format. The first FCCOB register, FCCOB0, always contains the command code. This 8-bit value defines the command to be executed. The command code is followed by the parameters required for this specific flash command, typically an address and/or data values.

NOTE

The command parameter table is written in terms of FCCOB Number (which is equivalent to the word number). This number is a reference to the FCCOB register name and is not the register address.

FCCOB Number[1]	Typical Command Parameter Contents [31:0]
0	FCMD [7:0] (flash command code, bits [31:8] are not writable)
1	FCMDOPT [7:0] (flash command options, bits [31:8] are not writable)
2	Flash address 1 (start)
3	Flash address 2 (end)
4	Data Word 0
5	Data Word 1
6	Data Word 2

Field	Function	
	FCCOB Number[1]	Typical Command Parameter Contents [31:0]
	7	Data Word 3
	1. Refers to FCCOB register name, not register address	

8.7 Glossary

Block Checker — MCU IP that provides a mechanism to set access rights to the flash memory for read and write operations.

Command write sequence — A series of MCU writes to the Flash FCCOB register group and FSTAT[CCIF] that initiates and controls the execution of flash algorithms that are built into the flash module.

Endurance — The number of times that an aligned flash or IFR phrase can be erased and reprogrammed.

FCCOB (Flash Common Command Object Block) — A group of flash registers that are used to pass command, address, data, and any associated parameters to the command controller.

Flash block — A macro within the flash module which provides the nonvolatile memory storage with an aligned block address being the first address in the block.

Flash module — All flash blocks plus a flash management unit providing command control and an interface to MCU buses.

Flash page — 128 bytes of flash main memory with an aligned page having byte-address[6:0] = 00h; represents the largest portion of the flash memory that can be programmed in one operation.

Flash phrase — 16 bytes of flash main memory with an aligned phrase having byte-address[3:0] = 0000b; represents the smallest portion of the flash memory that can be programmed in one operation.

Flash sector — 8 Kbytes of flash main memory with an aligned sector having byte-address[12:0] = 0000h; represents the smallest portion of the flash memory that can be erased in one operation.

FLW — The Flash Logical Window module allows a specific logical address range to access a programmable physical address range in the flash memory with programmable FLW default settings stored in IFR space.

FMC — Flash Memory Controller module manages flash memory reads and writes.

IFR — Information flash region separate from the main flash memory array. Each flash block has 32 Kbytes of user IFR space. Sometimes referred to as IFR0.

IFR1 — Information flash region separate from the main memory array reserved for array trim, MCU trim, and test. Each flash array has 8 Kbytes of IFR1 space.

IFR page — 128 bytes of IFR or IFR1 space with an aligned IFR page having byte-address[6:0] = 00h; represents the largest portion of IFR space that can be programmed.

IFR phrase — 16 bytes of IFR or IFR1 space with an aligned IFR phrase having byte-address[3:0] = 0000b; represents the smallest portion of IFR space that can be programmed in one operation.

IFR sector — 8 Kbytes of IFR or IFR1 space with an aligned IFR sector having byte-address[12:0] = 0000h; represents the smallest portion IFR space that can be erased.

MISR — Multiple-input signature register used to generate a signature based on flash memory contents read.

MSF — Microcontroller Secure Flash.

NVM — Nonvolatile memory. The flash block is an NVM using NOR-type flash memory technology.

Program flash — Program flash memory provides nonvolatile storage for vectors and code store.

Retention — The length of time that data (erased or programmed) can be kept in the NVM without experiencing errors upon readout.

RWW — Read-While-Write. The ability to simultaneously read from one flash memory resource while flash command operations are active in another flash memory resource.

Security/Privilege protection levels — The Flash module supports 4 levels of security/privilege protection for flash commands:

1. Secure (Trusted) / Privileged
2. Secure (Trusted) / User
3. Non-secure (Non-Trusted) / Privileged
4. Non-secure (Non-Trusted) / User

Chapter 9
Flash Controller (FMC)

9.1 Chip-specific FMC information

Table 70. Reference links to related information

Topic	Related module	Reference
Full description	FMC	FMC
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

See NVM Control (NVM_CTRL) for details on flash speculation and flash cache control.

NOTE

The Flash Controller module includes an optional PRINCE encryption/decryption features that is not described here. For details on this feature, refer to the MCX N Security RM.

9.1.1 ECC error check

The flash includes ECC checks on the memory contents. Any correctable, single-bit errors are automatically corrected. Non-correctable, double-bit errors are reported through the Error Recording Module (ERM). See the ERM chapter.

9.2 Overview

The Flash Memory Controller (FMC) is a memory interface and acceleration unit providing:

- An interface between the device and the nonvolatile memory
- Buffers that can accelerate flash memory transfers
  - A flash-phrase-sized buffer (128 bits) holds the most recently accessed flash phrase.
  - An optional flash-phrase-sized speculation buffer can prefetch the next flash phrase.
  - An optional 4-way, set-associative cache can store previously accessed flash phrases.

The FMC manages the interface between the device and the flash memory. The FMC receives status information describing the configuration of the memory and uses this information to ensure a proper interface. The following table shows the supported read and write operations.

Flash memory type	Read	Write
Program, IFR, IFR1 flash memory	8-bit, 16-bit, and 32-bit reads	16-byte and 128-byte writes

The FMC is controlled by a programmer's model external to the FMC module; see the chip-specific FMC information for details. The FMC's programming model provides a very configurable, high performance flexible memory controller, which can be optimized for the runtime characteristics of specific applications.

NOTE

Program the FMC's controls only while the flash controller is idle. Changing the configuration settings while a flash access is in progress can cause non-deterministic, unpredictable behavior.

9.2.1 Features

- Interface between the device and the flash memory:
  - The FMC's input bus supports 8-bit, 16-bit, and 32-bit read operations to flash memory.
  - The FMC's flash memory interface fetches a 128-bit flash phrase.
  - For input read requests, the FMC fetches a flash phrase with the desired read data from flash memory.
  - The flash memory interface can write aligned 16-byte flash write phrases or aligned 128-byte flash write pages to flash memory.
  - The FMC has a 16-byte aligned write buffer. This buffer is used once for aligned 16-byte flash write phrases or eight times for aligned 128-byte flash write pages.
  - The FMC's input bus supports 32-bit write operations for flash memory writes to fill the FMC's write buffer.
  - For input write requests, the FMC must receive the 4-word write of an aligned phrase in order.
- Acceleration of data transfer from flash memory to the device:
  - A flash-phrase-sized buffer that holds the current decrypted flash phrase fetched due to a FMC read request. Subsequent FMC read requests *that hit in the current buffer* return data with no wait states.
  - A flash-phrase-sized prefetch speculation buffer with controls for prefetching on instructions and/or data reads. When prefetching is enabled, idle FMC-to-flash interface cycles are used to fetch the next sequential flash phrase and hold it in the prefetch buffer. Subsequent FMC read requests *that hit in the speculation buffer* return data with no wait states.
  - Input controls:
    - to disable data type speculation
    - to disable all speculation
    - to invalidate the current and speculation buffers

    See the chip-specific section for details about controls.
  - The flash cache has input controls:
    - to disable instruction caching
    - to disable operand caching
    - to disable all caching
    - to clear the cache

    See the chip-specific section for details about controls.
  - The size of the flash cache in bytes is calculated as follows:

    flash cache size = [number of ways] × [number of sets] × [flash phrase size (in bytes)]

    For example, a flash cache with 4 ways, 1 set, and a 128-bit flash phrase (= 16 bytes) has a total flash cache size = 64 bytes

    (4 ways) × (1 set) × (16 bytes per flash phrase) = 64 bytes, the size of the flash cache

    NOTE

    Clear the speculation buffer and flash cache before accessing recently modified flash addresses. The flash cache has a specific clear control bit. To clear the speculation buffer, first disable then re-enable the speculation via other modules.

9.3 Functional description

The FMC is a flash interface and acceleration unit, with flexible buffers for user configuration.

- The FMC's input bus can operate faster than the flash memory.
- The FMC-to-flash interface has flow control to add wait states as needed (for input bus reads that need flash accesses).
- The FMC also contains various configurable buffers that hold recent flash accesses. If an input bus read hits a valid buffer, then that access will complete with no wait states.

9.3.1 Modes of operation

The FMC only operates when a bus master accesses the flash memory.

For any device power mode where the flash memory cannot be accessed, FMC is disabled automatically.

9.3.2 Default configuration

After system reset, the FMC is configured to provide a significant level of buffering for transfers from the flash memory. For all banks:

- The current and speculation buffers are cleared by reset.
- Prefetch support for data and instructions is enabled.
- The cache is cleared by reset.
- The cache is configured for data or instruction replacement.

9.3.3 Configuration options

The default configuration provides a high degree of flash acceleration, but advanced users may want to customize the FMC buffer configurations to maximize throughput for their use cases. When reconfiguring the FMC for custom use cases, do not program the FMC's control registers while the flash memory is being accessed. Instead, change the control registers with a routine executing from RAM in Supervisor mode.

The FMC's cache and buffering controls allow other modules to tune resources to suit specific application requirements. The cache and buffer are each controlled individually. The controls enable buffering and prefetching per memory bank and access type (instruction fetch or data reference).

See the chip-specific section for details about the registers used to configure FMC.

As an application example: if both instruction fetches and data references are accessing flash memory, then control is available to send instruction fetches, data references, or both to the cache or the single-entry buffer. Likewise, speculation can be enabled or disabled for either type of access.

In another application example, the cache can be configured for replacement from bank 0, while the single-entry buffer can be enabled for bank 1 only. This configuration is ideal for applications that use bank 0 for program space and bank 1 for data space.

For best performance, the FMC cache and speculation buffering should be enabled for instruction, data fetching, or both. The following is recommended for best performance:

- If the speculation buffer is enabled for both instruction and data speculation, enable the flash cache for both instruction and data caching.
- If the speculation buffer is enabled for instruction speculation only, enable the flash cache for at least instruction caching.

9.3.4 Wait states

Because the core, crossbar switch, and bus masters can be clocked at a higher frequency than the flash clock, flash memory accesses that do not hit in the speculation buffer or cache usually require wait states.

FMC does not allow the configuration of wait states directly. Wait states can be controlled via FMU FCTRL[RWSC].

9.3.5 Speculative reads

The FMC has a single buffer that reads ahead to the next phrase in the flash memory if there is an idle cycle. Speculative prefetching is programmable for instruction and data accesses. Because many code accesses are sequential, using the speculative prefetch buffer improves performance in most cases.

See the chip-specific section for information about controlling speculative reads.

When speculative reads are enabled, the FMC immediately requests the next sequential phrase address after a read completes. By requesting the next phrase immediately, speculative reads can help to reduce or even eliminate wait states when accessing sequential code and/or data.

9.3.6 Interrupts

This module has no interrupts.

9.4 External signals

The FMC has no external signals.

9.5 Initialization and application information

The FMC does not require user initialization. Flash acceleration features are enabled by default.

The FMC has no visibility into flash memory erase and program cycles because the Flash Memory module manages them directly. To prevent the possibility of returning stale data when an application is executing flash memory commands, disable and/or flush FMC's current buffer, speculation buffer, and cache. While you can disable these features individually, we recommend disabling all of them when executing flash memory commands.

See the chip-specific section for details about the registers used to disable features.

Chapter 10
FlexSPI

10.1 Chip-specific FlexSPI information

Table 71. Reference links to related information

Topic	Related module	Reference
Full description	FlexSPI	FlexSPI
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

10.1.1 Module instances

This device has one instance of the FlexSPI module, FLEXSPI0.

NOTE

The FlexSPI module includes an optional IPED GCM feature that is not described here. For details on this feature, refer to the MCX N Security RM.

10.1.2 FlexSPI AHB regions

There are multiple FlexSPI regions in the system memory map, but the data accessed is aliased across the multiple regions. Refer to System memory map to see FlexSPI AHB spaces on this device. Also, refer to FlexSPI regions mapping to CACHE64_CTRL and CACHE64_POLSEL.

10.1.3 Controller ID allocation

The Controller ID (MSTRID) values used for the FlexSPI module are same as the AHB bus matrix port numbers. Refer to Table 2 to find the controller port/MSTRID value used for each controller.

10.1.4 FlexSPI connection modes

The table below shows details about the Parallel and Individual modes for flash memory.

Table 72. Relation between modes and bus size for FlexSPI instances

Mode	Effective Bus Size	Data Signals	Function	Devices Attached to Bus	Chip Selects
INDIVIDUAL	4 bit	A_DATA0- A_DATA3	Single QSPI chip support(1b,2b,4b)	Up to 2	A_SS0_B, A_SS1_B
INDIVIDUAL	4 bit	B_DATA0- B_DATA3	Single QSPI chip support(1b,2b,4b)	Up to 2	B_SS0_B, B_SS1_B
INDIVIDUAL	8 bit	A_DATA0-	Single Octal/	Up to 2	A_SS0_B,

Table continues on the next page...

Table 72. Relation between modes and bus size for FlexSPI instances (continued)

Mode	Effective Bus Size	Data Signals	Function	Devices Attached to Bus	Chip Selects
		A_DATA7	Hyperbus/Xccela chip support(1x8b)		A_SS1_B
INDIVIDUAL	8 bit	B_DATA0- B_DATA7	Single Octal/ Hyperbus/Xccela chip support(1x8b)	Up to 2	B_SS0_B, B_SS1_B
PARALLEL	8 bit	A_DATA0- A_DATA3 | B_DATA0- B_DATA3	Two QSPI chips in parallel(2x4b)	2 parallel chips x2 = up to 4	(A_SS0_B | B_SS0_B), (A_SS1_B | B_SS1_B)
PARALLEL	16 bit	A_DATA0- A_DATA7 | B_DATA0- B_DATA7	Two Octal/ Hyperbus/Xccela chips in parallel(2x8b)	2 parallel chips x2 = up to 4	(A_SS0_B | B_SS0_B), (A_SS1_B | B_SS1_B)

10.2 Overview

FlexSPI supports two SPI channels and up to four external devices. Each channel supports Single/Dual/Quad/Octal mode data transfer (1/2/4/8 bidirectional data lines).

The FlexSPI configuration depends on the chip configuration. See the system-level section for boot information and pinmux for chip-specific information regarding the modes and number of devices supported.

FlexSPI supports communication with both serial flash memory and serial RAM devices. While many of the descriptions, registers, and fields specifically reference flash memory, almost all information can also be applied to serial RAM. Flash memory is used as an example in the tables and figures in this chapter.

NOTE

Terminology in this chapter has been updated to align with JEDEC standard *Expanded Serial Peripheral Interface (xSPI) for Non Volatile Memory Devices, Version 1.0*.

Table 73. Updated terms

Updated term	Deprecated term
Controller	Master
Target	Slave

10.2.1 Block diagram

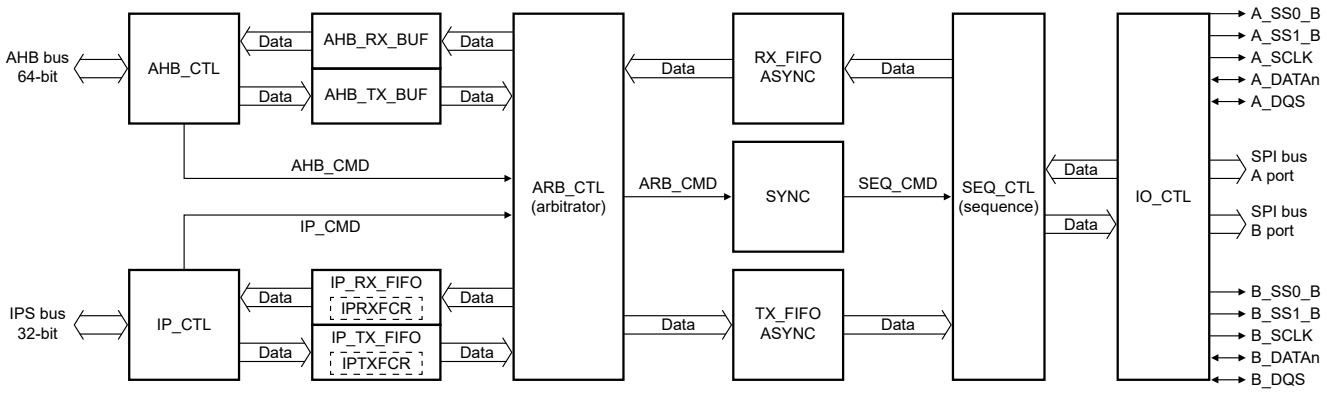


Figure 11. Block diagram

10.2.2 Features

FlexSPI supports:

- Flexible sequence engine (lookup table) to support various vendor devices
  - Serial NOR flash memory and other devices with SPI protocol similar to serial NOR flash memory
  - Serial NAND flash memory
  - HyperBus devices (HyperFlash/HyperRAM)
  - XCCELA devices
  - FPGA devices
- Flash memory access modes
  - Single, Dual, Quad, Octal mode
  - Single Data Transfer Rate (SDR) and Double Data Transfer Rate (DDR) mode
  - Individual/Parallel mode
- Sampling clock mode
  - Internal dummy read strobe looped back internally
  - Internal dummy read strobe looped back from pad
  - SCLK clock output looped back from pad
  - Flash-memory-provided read strobe
- Automatic data learning to select the correct sample clock phase
- Memory mapped read and write access by AHB bus
  - AHB receive buffer implemented to reduce read latency. Total AHB receive buffer size: 1024 bytes.
  - 16 AHB controllers with programmable priority for read access from each
  - 8 flexible and configurable buffers in AHB receive buffer
  - AHB transmit buffer implemented to buffer all write data from one AHB burst. AHB transmit buffer size is 64 bytes.

NOTE

All AHB controllers share this AHB transmit buffer. There is no AHB controller number limitation for write access.

- Software-triggered flash memory read and write access by IP bus
  - IP receive FIFO implemented to buffer all read data from external devices. FIFO size: 1024 bytes
  - IP transmit FIFO implemented to buffer all write data to external devices. FIFO size: 1024 bytes
  - DMA support to read IP receive FIFO
  - DMA support to fill IP transmit FIFO
  - SCLK stops when IP receive FIFO is full during reading flash memory data
  - SCLK stops when IP transmit FIFO is empty during writing flash memory data

10.3 Functional description

10.3.1 Flash connection

There are two FlexSPI interface ports (port A and port B). Each port supports up to two flash devices by providing two chip select outputs.

NOTE

FlexSPI configuration depends on the chip configuration. See the chip-specific FlexSPI information regarding the number of devices supported.

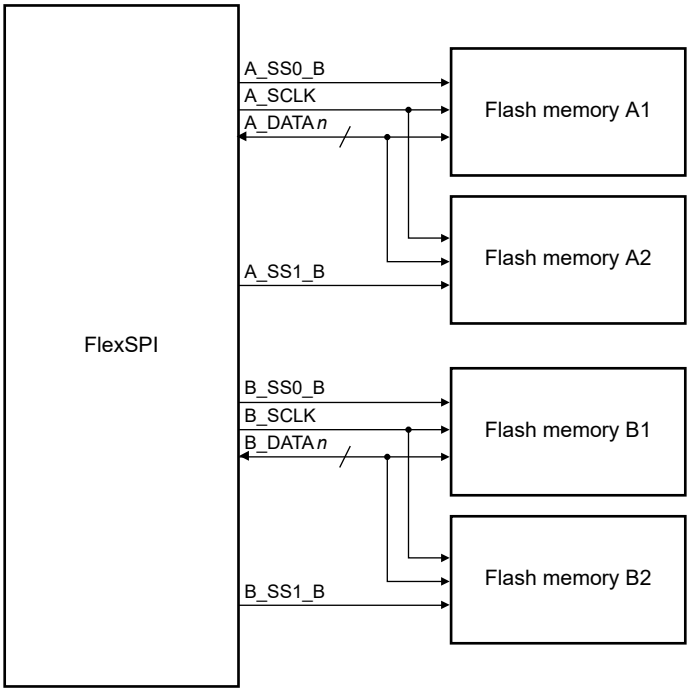


Figure 12. FlexSPI connection diagram with four devices.

NOTE

- Flash A1 and Flash A2 can be two flash chip packages or two flash dies in the same package. There is no difference to FlexSPI. This statement is also true for Flash B1 and Flash B2.
- Flash A1 and Flash B1 can be accessed in parallel, using parallel mode. FlexSPI merges or splits the flash read and program data automatically. This statement is also true for Flash A2 and Flash B2.
- In parallel mode, Flash A1 and Flash A2 cannot be accessed at the same time. This statement is also true for Flash B1 and Flash B2.
- In individual mode, flash devices cannot be accessed at the same time. But these four devices can be accessed separately.

There is a combination mode (MCR0[COMBINATIONEN] = 1) used to provide octal flash memory support by combining Port A (A_DATA[3:0]) and Port B (B_DATA[3:0]) together. Normal mode can also support Octal mode by using all eight data lines from Port A or Port B. Figure 13 shows the connection diagram for combination mode.

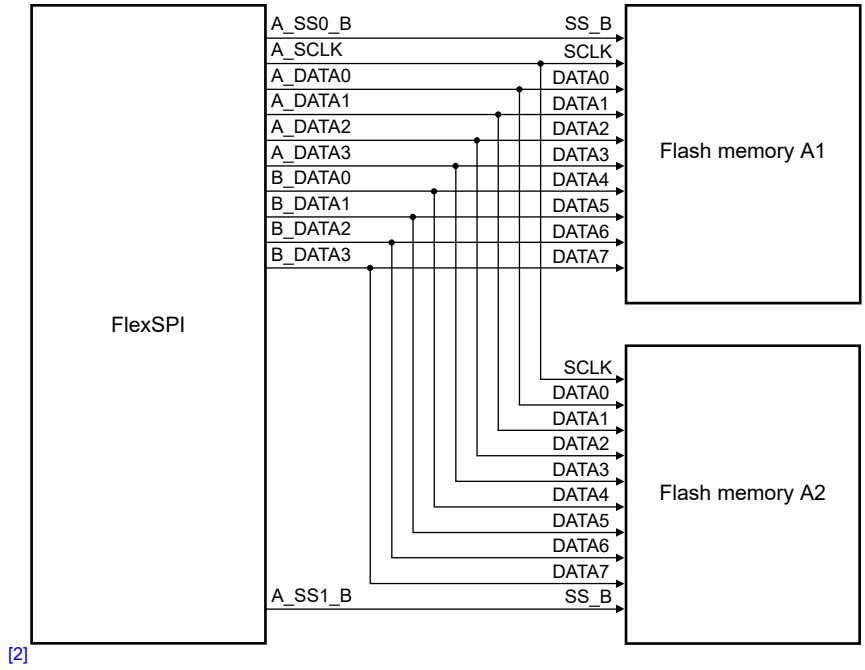


Figure 13. FlexSPI connection diagram for combination mode

10.3.2 Flash memory access mode

10.3.2.1 SPI clock mode

FlexSPI only supports SPI clock mode 0: clock polarity (CPOL) = 0 and clock phase (CPHA) = 0. When the SPI bus is idle, SCLK stays at a logic low state.

10.3.2.2 Individual mode and Parallel mode

In individual mode, read and write data are received and transmitted on port A or port B.

In parallel mode, read and write data are received and transmitted on port A and port B in parallel. FlexSPI merges or splits the read and program data automatically (READ/WRITE instruction). Only read and program data is merged or split (READ/WRITE

[2] Parallel mode access is not available in combination mode.

instruction). For other instructions (such as Command, Address, Mode, and Data size), the same associated information is transmitted to the devices connected to port A and port B. For more details, see Executing instructions on SPI interface.

IPCR1[IPAREN] (for IP command) or AHBCR[APAREN] (for AHB command) statistically determines Individual and Parallel mode.

NOTE

FlexSPI does not support 16-bit read and write for the SPI interface.

10.3.2.3 SDR mode and DDR mode

In SDR mode, flash memory receives data on the rising edge of SCLK and transmits data on the falling edge of SCLK.

In DDR mode, flash memory receives and transmits data on both the rising and falling edges of SCLK.

The instruction (opcode) in the LUT sequence dynamically determines SDR and DDR modes. There is no static configuration register field setting for SDR and DDR modes. See FlexSPI input timing and FlexSPI output timing for details about input and output timing.

10.3.2.4 Single, Dual, Quad, and Octal mode

In Single mode, FlexSPI transmits and receives data on one data pin (DATA0 for transmitting, DATA1 for receiving).

In Dual mode, FlexSPI transmits and receives data on two data pins (DATA0–DATA1 for both transmitting and receiving).

In Quad mode, FlexSPI transmits and receives data on four data pins (DATA0–DATA3 for both transmitting and receiving).

In Octal mode, FlexSPI transmits and receives data on eight data pins (DATA0–DATA7 for both transmitting and receiving).

Instruction (num_pads) in the LUT sequence dynamically determine Single, Dual, Quad, and Octal modes. There is no static configuration register field setting for Single, Dual, Quad, and Octal modes.

10.3.3 Modes of operation

FlexSPI operates with these modes.

Table 74. Operating modes

Mode	Description
Module Disable	This mode is a low-power mode for FlexSPI. In Module Disable mode, the AHB clock and serial clock domains are gated off internally, but the IPS bus clock domain is not gated off. Read and write access to the control and status register are available, but the LUT, IP receive FIFO, and IP transmit FIFO cannot be accessed. Serial memory access is also not available. Write 1 to MCR0[MDIS] to enter this mode. Write 0 to MCR0[MDIS] to exit this mode.
Doze	This mode is a low-power mode for the chip. When the chip requires FlexSPI to enter Doze mode and MCR0[DOZEEN] = 1, FlexSPI enters Doze mode after all transactions are completed (STS0[ARBIDLE] = 1). In Doze mode, the AHB clock and serial clock domains are gated off internally, but the IPS bus clock is not gated off. Read and write access to the control and status register are available, but the LUT, IP receive FIFO, the IP transmit FIFO cannot be accessed. Serial memory access is also not available. This mode is entered via system request, and exited by deasserting this system request.
Stop	This mode is a low-power mode for the chip.

Table continues on the next page...

Table 74. Operating modes (continued)

Mode	Description
	When the chip requires the FlexSPI to enter Stop mode, FlexSPI waits for all transactions to complete (STS0[ARBIDLE] = 1) and return ACK handshake to the system. After the ACK handshake is returned, FlexSPI gates off the AHB clock and serial clock domains internally. The system can gate off the AHB bus clock, IPS bus clock, and serial clock at the system level. This mode is entered via system request. This mode is exited by deasserting this system request, and the ACK handshake message is also deasserted immediately.
Normal	No clock is gated off internally. Normal register access and serial memory access are available.

10.3.4 AHB access memory map

FlexSPI allocates AHB memory space for each memory device starting from the FlexSPI region base address of the system memory map. The FLSHxCR0[FLSHSZ] field determines the amount of memory allocated for each memory device in KB, Fx_SIZE. The memory is then allocated sequentially and contiguously.

NOTE

The maximum flash size supported for each device is 512 MB. The maximum total flash size supported (for all four devices) is also 512 MB.

When MCR2[SAMEDEVICEEN] = 1:

- FA1_SIZE = FLSHA1CR0[FLSHSZ] × 1 KB
- FA2_SIZE = FLSHA1CR0[FLSHSZ] × 1 KB
- FB1_SIZE = FLSHA1CR0[FLSHSZ] × 1 KB
- FB2_SIZE = FLSHA1CR0[FLSHSZ] × 1 KB

When MCR2[SAMEDEVICEEN] = 0:

- FA1_SIZE = FLSHSZ × 1 KB
- FA2_SIZE = FLSHA2CR0[FLSHSZ] × 1 KB
- FB1_SIZE = FLSHB1CR0[FLSHSZ] × 1 KB
- FB2_SIZE = FLSHB2CR0[FLSHSZ] × 1 KB

Flash B1 and Flash B2 size settings are ignored in parallel mode (FLSHB1CR0[FLSHSZ], FLSHB2CR0[FLSHSZ]). To execute in parallel mode, Flash B1 should be the same device as Flash A1, and Flash B2 should be the same device as Flash A2.

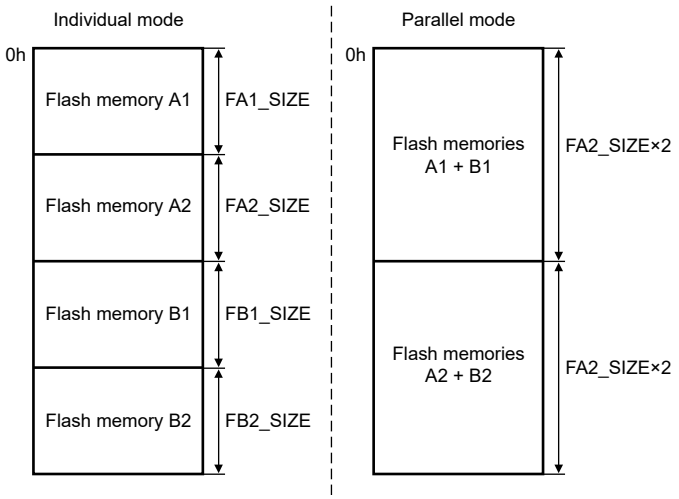


Figure 14. AHB access memory map in individual and parallel mode

AHB access memory map in individual mode:

- Flash A1 address range: 0000_0000h to FA1_SIZE
- Flash A2 address range: FA1_SIZE to (FA1_SIZE + FA2_SIZE)
- Flash B1 address range: (FA1_SIZE + FA2_SIZE) to (FA1_SIZE + FA2_SIZE + FB1_SIZE)
- Flash B2 address range: (FA1_SIZE + FA2_SIZE + FB1_SIZE) to (FA1_SIZE + FA2_SIZE + FB1_SIZE + FB2_SIZE)

AHB access memory map in parallel mode:

- Flash A1+B1 address range: 0000_0000h to FA1_SIZE × 2
- Flash A2+B2 address range: FA1_SIZE × 2 to (FA1_SIZE × 2 + FA2_SIZE × 2)

NOTE

The address used in Figure 14 and equations in this topic is the address presented to the memory. The base address for FlexSPI in the system memory map has already been removed, so it not used or shown here.

10.3.4.1 Dual image use case using HADDRSTART, HADDREND, and HADDROFFSET registers

The FlexSPI controller has a remap feature that supports the storage of dual images in memory. The HADDRSTART and HADDREND registers can be used to configure an address range to which an offset (that the HADDROFFSET register configures) is applied. This feature allows a single system bus address range to be mapped to two different memory ranges: the range that is remapped or the range that is not remapped. You can enable and disable the remapping to switch between accessing the two images at the same system memory address. This feature is used for booting.

NOTE

- Remapping in this table includes the REMAP register (HADDRSTART[0]) and SWAP register (HADDRSTART[2]).
- The swap function impacts both the AHB and IPS address remapping with the same mechanism. The remap function only impacts the AHB address.

Table 75. FlexSPI address translation

	AHB or IPS address	Memory address	Description
Enable remap			

Table continues on the next page...

Table 75. FlexSPI address translation (continued)

	AHB or IPS address	Memory address	Description
AHB WR	ADDR	ADDR + OFFSET	The remap function is available if HADDRSTART <= ADDR < HADDREND, otherwise no change in address.
AHB RD	ADDR	ADDR + OFFSET	The remap function is available if HADDRSTART <= ADDR <HADDREND, otherwise no change in address.
Disable remap			
AHB WR	ADDR	ADDR	No change in address.
AHB RD	ADDR	ADDR	No change in address.

NOTE

- HADDRSTART[0] is used to enable the remap feature.
- The ADDR above is the address from the IPS or AHB interface. The OFFSET is same as the HADDROFFSET register.

10.3.4.2 Flash address sent to flash memory devices

The AHB address (AHB command) or IPCR0[SFAR] (IP command) determines the flash memory access start address. See Flash memory access via AHB command and Flash memory access via IP command for more details.

For AHB commands, the FlexSPI controller removes the flash memory base address automatically when sending flash addresses to flash devices. The flash address is sent to devices in two parts: row address and column address. For flash devices that do not support the column address, set FLSHxCR1[CAS] to 0. This setting causes all flash address bits to be sent to flash devices as row addresses.

For word-addressable flash devices, the last bit of the address is not needed, because flash memory is read and programmed in terms of two bytes. For parallel mode, Flash A1 and Flash B1 (or Flash A2 and Flash B2) are accessed in parallel. As a result, the flash memory address sent to flash devices should be divided by 2. Table 76 indicates the relationship of row address, column address, and flash address (FA).

Table 76. Flash address, row address, and column address

Parallel mode	Word-addressable	Row address	Column address	Comment
0	0	FA[31:CAS]	FA[CAS-1:0]	There is no limitation on FA and data size alignment.
0	1	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be two-byte aligned.
1	0	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be two-byte aligned.
1	1	FA[31:CAS+2]	FA[CAS+1:2]	FA and data size should be four-byte aligned.

NOTE

- FA is the flash address without base address.
- If the bit number of the row or column flash address is more than the valid value, the high position bits are supplemented with zero. See Programmable sequence engine for details about row or column address instructions.

When parallel mode is enabled or word-addressable flash memory is used, there are limitations on the flash memory start address and data size. You can address this requirement by aligning the AHB bus access address (for AHB command) or IP command address IPCR0[SFAR] (for IP command) in software. There are two ways to avoid these limitations in the specified case.

1. For AHB Read Command only:

   When AHBCR[READADDROPT] and AHBCR[PREFETCHEN] are both 1, FlexSPI guarantees the start address and data size for the flash memory access are eight-byte aligned by hardware.

   When AHBCR[READADDROPT] is 1, FlexSPI fetches redundant data to guarantee the flash memory start address is eight-byte aligned. When prefetch is enabled (AHBCR[PREFETCHEN] is 1), the size of the eight-byte-aligned AHB receive buffer determines the flash memory read data size.

2. For AHB Write Command and Individual Mode only:

   By default, FlexSPI guarantees the flash memory write access start address and data size are two-byte aligned when using DQS as write mask (FLSHCR4[WMENB or WMENA] = 1).

   This feature is not applied in parallel mode and must be used if the external device supports write mask feature.

   For example, for odd start address:

   - To write 6 bytes, FlexSPI sends 8 bytes, the first and last byte are both masked.
   - To write 7 bytes, FlexSPI sends 8 bytes, only the first byte is masked.
   - To write 8 bytes, FlexSPI sends 10 bytes, the first and last byte are both masked.

   For even start address:

   - To write 6 bytes, FlexSPI sends 6 bytes, no bytes are masked.
   - To write 7 bytes, FlexSPI sends 8 bytes, last byte is masked.
   - To write 8 bytes, FlexSPI sends 8 bytes, no bytes are masked.

10.3.5 Lookup table (LUT)

The LUT is an internal memory that preserves a number of preprogrammed sequences. Each sequence consists of up to eight instructions which are executed sequentially. When an IP command or an AHB command triggers a flash memory access, the FlexSPI controller:

1. Fetches the sequence from LUT (sequence index or number).
2. Executes the flash memory access to generate a valid flash transaction on the SPI interface.

Figure 15 shows the LUT structures, sequences, and instructions.

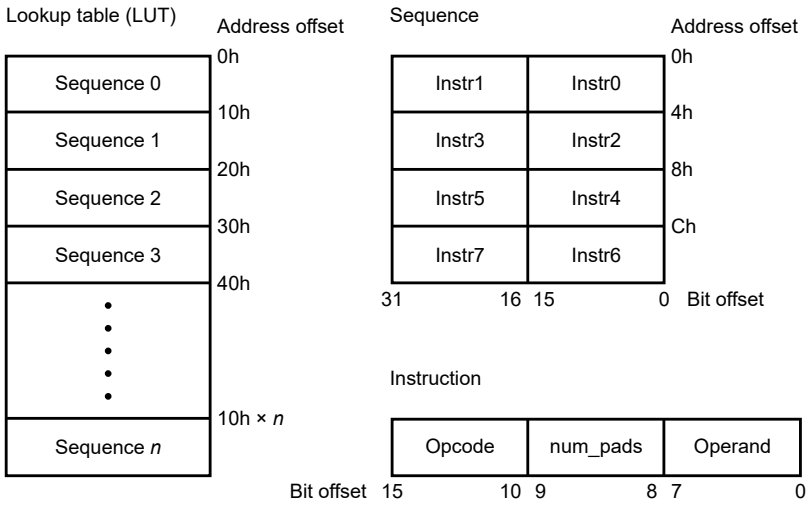


Figure 15. LUT structure, sequences, and instructions

NOTE

If the number of instructions needed for a flash transaction is less than eight, the STOP instruction (code 0000_0000h) must be programmed for unneeded instructions.

For IP and AHB write commands, the FlexSPI controller always executes from instruction pointer 0. For AHB read commands, the FlexSPI controller executes from a saved instruction start pointer. The FlexSPI controller saves the instruction start pointers separately for each flash device. All these saved instruction pointers are zero before the JMP_ON_CS instruction is executed. When the JMP_ON_CS instruction is executed, the operand in the JMP_ON_CS instruction is saved as instruction start pointer. See Execute-In-Place (XIP) enhanced mode.

The reset value of the LUT is unknown because it is implemented as internal memory. The LUT must be programmed according to the connected device. To protect its contents during a code runover, the LUT can be locked or unlocked to prevent unwanted changes after the LUT has been configured. The key to lock or unlock the LUT is **5AF05AF0h**.

To lock the LUT:

1. Write the key (5AF05AF0h) to LUT Key (LUTKEY).
2. Write 1 to LUTCR[LOCK] and write 0 to LUTCR[UNLOCK]. When there is another register write access to FlexSPI between these two write accesses, this LUT is not successfully locked.

To unlock the LUT:

1. Write the key (5AF05AF0h) to the LUT Key Register LUT Key (LUTKEY).
2. Write 0 to LUTCR[LOCK] and write 1 to LUTCR[UNLOCK]. When there is another register write access to FlexSPI between these two write accesses, this LUT is not successfully locked.

The lock status of the LUT can be read from LUTCR[LOCK] and LUTCR[UNLOCK].

10.3.6 Programmable sequence engine

The FlexSPI controller implements a programmable sequence engine that executes the sequence from the LUT. The FlexSPI controller executes the instructions sequentially, and generates flash transactions on the SPI interface accordingly. Table 77 is a complete list of the supported instructions.

Table 77. Instruction set

Name	Opcode	num_pads	Action on SPI interface	Transmit data	Bits/bytes/cycle number
CMD_SDR	00_0001h	00h - one pad (Single mode) 01h - two pads (Dual mode) 02h - four pads (Quad mode) 03h - eight pads (Octal mode)	Transmit command code to flash memory	Command code: Operand[7:0]	Bit number: 8
CMD_DDR	00_0021h				
RADDR_SDR	00_0002h		Transmit row address to flash memory See Flash address sent to flash memory devices	Row_Address[31:0] Row_Address comes from AHB bus (AHB command) or IPCR0[SFAR] (IP command).	Bit number: operand[7:0] Value of operand determines number of bits sent in Row_Address.
RADDR_DDR	00_0022h				
CADDR_SDR	00_0003h		Transmit column address to flash memory. See Flash address sent to flash memory devices.	Column_Address[31:0] Column_Address comes from AHB bus (AHB command) or IPCR0[SFAR] (IP command).	Bit number: operand[7:0] Value of operand determines number of bits sent in Column_Address.
CADDR_DDR	00_0023h				
MODE1_SDR	00_0004h		Transmit mode bits to flash memory	Mode bits: Operand[0]	Bit number: 1
MODE1_DDR	00_0024h				
MODE2_SDR	00_0005h			Mode bits: Operand[1:0]	Bit number: 2
MODE2_DDR	00_0025h				
MODE4_SDR	00_0006h			Mode bits: Operand[3:0]	Bit number: 4
MODE4_DDR	00_0026h				
MODE8_SDR	00_0007h			Mode bits: Operand[7:0]	Bit number: 8
MODE8_DDR	00_0027h				
WRITE_SDR	00_0008h		Transmit program data to flash device	Program data in IP_TX_FIFO or AHB_TX_BUF	AHB burst size and burst type (AHB command) or IPCR1[IDATSZ] (IP command) determines byte number (data size). For details about flash read or program data size, see Flash memory access via AHB command and Flash memory access via IP command.
WRITE_DDR	00_0028h				
READ_SDR	00_0009h		Receive read data from flash device Read data is put into AHB_RX_BUF or IP_RX_FIFO.	-	
READ_DDR	00_0029h				
DATSZ_SDR	00_000Bh		Transmit read or program data size	Internal logic	Bit number: operand[7:0]
DATSZ_DDR	00_002Bh				

Table continues on the next page...

Table 77. Instruction set (continued)

Name	Opcode	num_pads	Action on SPI interface	Transmit data	Bits/bytes/cycle number
			(byte number) to flash device	Read or program data size for current command sequence	Never set operand to zero or greater than 64 for DATSZ instruction.
DUMMY_SDR	00_000Ch		Leave data lines undriven by FlexSPI controller. Turnaround cycles are provided from host driving to device driving. num_pads determines number of pads in input mode.	-	Dummy cycle number (in serial root clock): Operand[7:0] Dummy cycle (N), described in data sheet of flash device, is in number of SCLK cycles. This number may be configurable. In SDR mode, SCLK cycle is same as serial root clock. Operand value must be set to N. In DDR mode, SCLK cycle is double serial root clock cycle. Operand value must be set to 2N, 2N-1 or 2N+1 depending on definition of dummy in data sheet of flash device. See Flash memory access sequence examples and dummy cycle definition in data sheet of external memory.
DUMMY_DDR	00_002Ch				
DUMMY_RWDS_SDR	00_000Dh		Similar to DUMMY_SDR/ DUMMY_DDR instruction. Difference is in dummy cycle number. DQS pin is called RWDS in HyperBus specification. See Dummy instruction for details. **Set operand to "Latency count" for HyperBus devices.**	-	For read command, dummy cycle number (in serial root clock): (operand[7:0] × 4 - 1) if RWDS (DQS pin) is high; (operand[7:0] × 2 - 1) if RWDS (DQS pin) is low; For write command, dummy cycle number (in serial root clock): (operand[7:0] × 4 - 2) if RWDS (DQS pin) is high; (operand[7:0] × 2 - 2) if RWDS (DQS pin) is low;
DUMMY_RWDS_DDR	00_002Dh				

Table continues on the next page...

Table 77. Instruction set (continued)

Name	Opcode	num_pads	Action on SPI interface	Transmit data	Bits/bytes/cycle number
LEARN_SDR	00_000Ah	00h - one pad (Single mode) 01h - two pads (Dual mode) 02h - four pads (Quad mode) 03h - eight pads (Octal mode)	Receive read data or preamble bit from flash device FlexSPI controller compares data line bits with the DLPR register to determine a correct sampling clock phase.	-	Bit number: operand[7:0] Never set operand to zero for LEARN instruction. Value of operand indicates number of bits to receive and compare to DLPR value. For example, 8-bit pattern 5Ah on each data line must set operand to 8.
LEARN_DDR	00_002Ah				
JMP_ON_CS	00_001Fh	Num_pads setting ignored.	Stop execution, deassert CS and save operand[7:0] as instruction start pointer for next sequence. Normally this instruction is used to support Execute-In-Place enhanced mode. See Execute-In-Place (XIP) enhanced mode. This instruction is only allowed for AHB read commands. When using this instruction in IP command or AHB write command, interrupt status flag set (INTR[IPCMDERR] or INTR[AHBCMDERR]).	-	No transaction on SPI interface.
STOP	00_0000h		Stop execution and deassert CS. Next command sequence (to same flash device) starts from instruction pointer 0.	-	

The programmable sequence engine allows configuration of the LUT according to the connected external serial device. The flexible LUT structure easily adapts to new command or protocol changes from different vendors.

The DDR sequence is a flash memory access sequence that contains DDR instructions other than DUMMY_DDR, and may contain SDR instructions. The output and input timing on FlexSPI differs for SDR and DDR sequences. When included as part of a DDR sequence, SDR instructions execute differently from SDR instructions in an SDR sequence. See FlexSPI input timing and FlexSPI output timing.

10.3.6.1 Executing instructions on SPI interface

This section describes the execution of instructions on the SPI interface. For all instructions that receive bits from or transmit bits to flash devices:

- The bit order in one byte is higher on DATA7 than DATA0.
- The bit order is higher on port B than port A.

This order is shown in Figure 16.

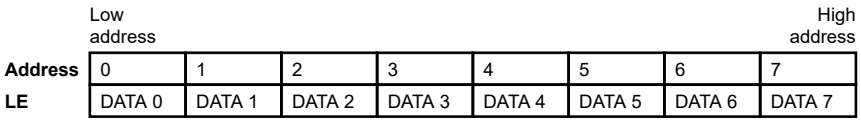


Figure 16. Bus bit order

Table 78. SPI instructions

Instructions	Description
Command (CMD_SDR, CMD_DDR)	Normally used to transmit a command code to the external device. The command code is the 8-bit operand in the instructions. Command code is sent to both port A and port B in parallel mode. See Flash memory access sequence examples.
Address (RADDR_SDR, RADDR_DDR, CADDR_SDR, CADDR_DDR)	Normally used to send the flash memory access start address (row address or column address) to an external device. FlexSPI determines the bits of the row address or column address according to the AHB access address or IP command address. See Flash address sent to flash memory devices. The operand value in the instruction code represents the number of address bits to send. The bits in the row address or column address are sent to both port A and port B in parallel mode. For example, when NUM_PADSn = 8, a memory reading the flash memory address on four SCK edges requires the sum of bits in CADDR and RADDR instructions to be 32. Otherwise, FlexSPI does not generate four SCK edges of address phase. See Flash memory access sequence examples.
Mode (MODEx_SDR, MODEx_DDR)	Normally used to send mode bits to external devices. Mode bits are the lower bits of an operand. For example, the bit number is 1 for MODE1_SDR and MODE1_DDR, 2 for MODE2_SDR and MODE2_DDR, 4 for MODE4_SDR and MODE4_DDR, and 8 for MODE8_SDR and MODE8_DDR. The pad number should not be greater than the mode bit number. For example, you cannot set NUM_PADSn to 11b (Octal mode) for MODE4_* instructions. Mode bits are sent to both port A and port B in parallel mode. See Flash memory access sequence examples for more details.
Data Size (DATSZ_SDR, DATSZ_DDR)	Used to send the size of program data or read data (byte number) to external devices. This instruction is normally used in FPGA applications where the memory space in external device acts like a FIFO. The external device requires data size information to determine how much

Table continues on the next page...

Table 78. SPI instructions (continued)

Instructions	Description
	data are popped from or pushed into the internal FIFO. The operation value in the data size instructions is the bit number. Data size is sent to both port A and port B in parallel mode. See Flash memory access sequence examples.
Write (WRITE_SDR, WRITE_DDR)	Normally used to send program data to external device. Programming data is fetched from IP_TX_FIFO (IP Command) or AHB_TX_Buffer (AHB command). For details about flash program data size, see Flash memory access via AHB command and Flash memory access via IP command. The byte order for program data is always from low to high. Odd bytes are sent on port A and even bytes are sent on port B in parallel mode. See Flash memory access sequence examples for more details.
Read (READ_SDR, READ_DDR)	Normally used to receive flash memory data from external devices. Received data is put into IP_RX_FIFO (IP Command) or AHB_RX_Buffer (AHB command). For information about flash memory read data size, see Flash memory access via AHB command and Flash memory access via IP command. The byte order for reading data is always from low to high. Odd bytes are received from port A and even bytes are received from port B in parallel mode. See Flash memory access sequence examples.
Dummy (DUMMY_SDR, DUMMY_DDR, DUMMY_RWDS_SDR, DUMMY_RWDS_DDR)	Used to provide turnaround cycles on the SPI interface. During dummy instruction, the FlexSPI controller and external devices do not drive the SPI interface. See Programmable sequence engine for details about dummy cycle number. DUMMY_RWDS_DDR can be used for a HyperBus device that uses RWDS pin to indicate whether extra latency is needed. DUMMY_RWDS_SDR is reserved. The FlexSPI controller checks the DQS pin input level at the fourth cycle after SCLK output toggling is enabled. The DQS pin is called RWDS in HyperBus specification. See Flash memory access sequence examples. **NOTE** FlexSPI releases the bus after at least one cycle. To avoid data contention, the NUM_PADSn value for the dummy commands should be configured to match the number of data lines used for the external memory.
Learn (LEARN_SDR, LEARN_DDR)	Used to determine the correct sample clock phase for flash memory read data sampling. External device drives read data (or data learning pattern) bits on the FlexSPI interface. The FlexSPI controller compares the data line bits to the DLPR register to determine a correct sampling clock phase. Clock phase selection is automatically updated after executing learn instructions. See Data learning for more details. The operand value in learn instructions is the byte number. FlexSPI checks the same data pattern on each data line. The byte order is byte 0, byte 1, byte 2, byte 3, byte0, byte 1, and so on. • Byte 0 is DLPR register bits 7–0. • Byte 1 is DLPR register bits 15–8. • Byte 2 is DLPR register bits 23–16. • Byte 3 is DLPR register bit 31–24. The bit order is from high to low in each byte. See Flash memory access sequence examples.

10.3.6.2 Flash memory access sequence examples

Diagrams below all assume MCR0[SERCLKDIV] = 0.

Figure 17 shows an example SDR single I/O read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

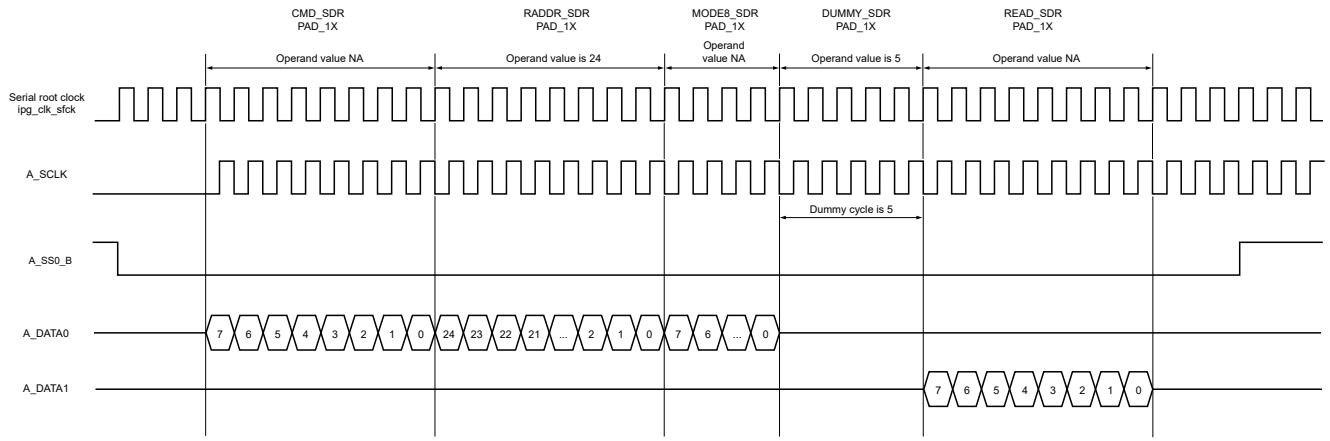


Figure 17. Flash memory access sequence example (SDR single I/O read sequence)

NOTE

- FlexSPI dummy instruction starts and ends at the rising edge of serial root clock.
- Device dummy cycle starts and ends at the falling edge of SCLK (on Cypress S25FS512S data sheet).

Figure 18 shows an example SDR quad I/O read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

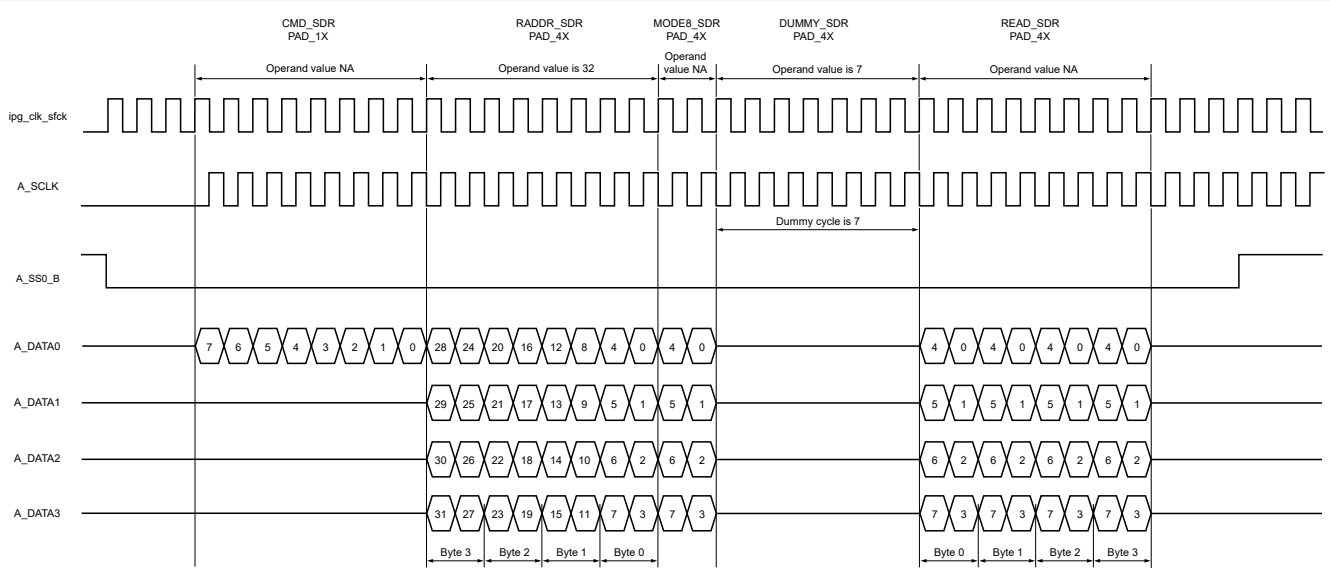


Figure 18. Flash memory access sequence example (SDR quad I/O read sequence)

NOTE

- FlexSPI dummy instruction starts and ends at the rising edge of serial root clock.
- Device dummy cycle starts and ends at the falling edge of SCLK (on Cypress S25FS512S data sheet).

Figure 19 shows an example DDR quad I/O read sequence (Cypress Serial Nor Flash S25FS512S) in parallel mode.

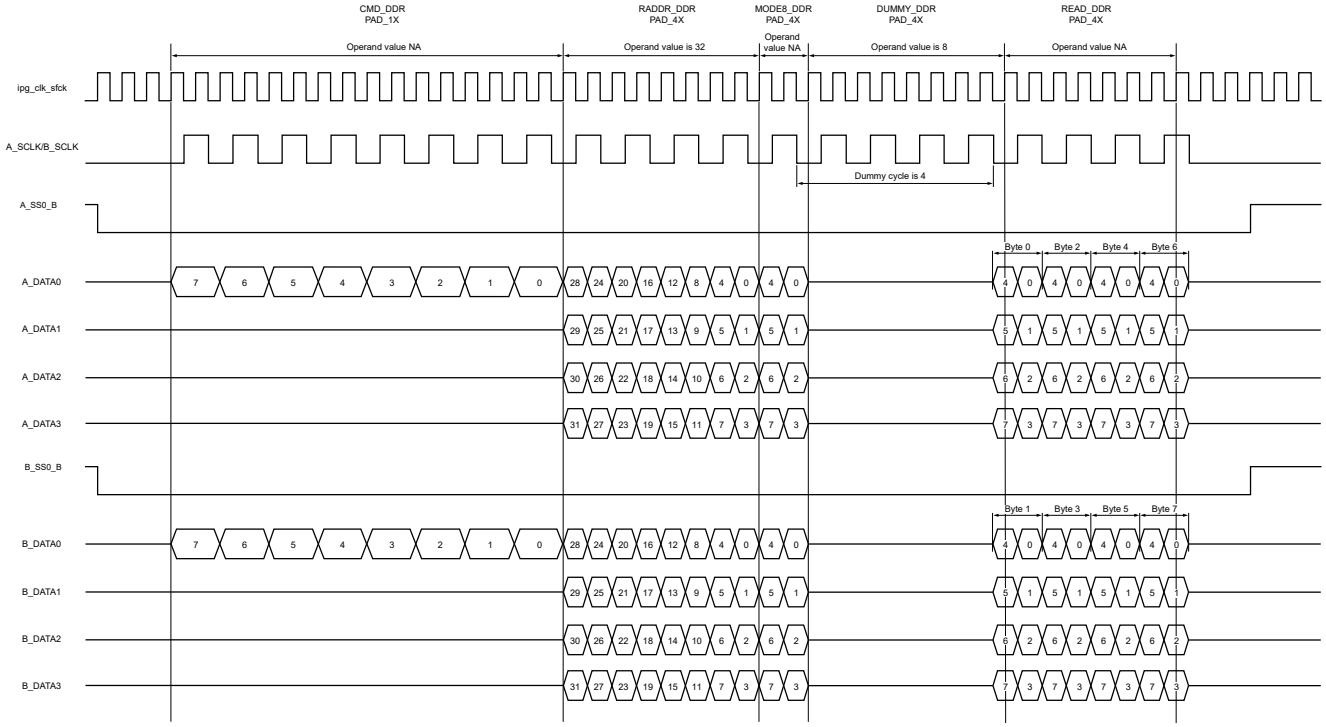


Figure 19. Flash memory access sequence example (DDR quad I/O read sequence)

Figure 20 shows an example learn instruction (not for a specified flash device) in individual mode.

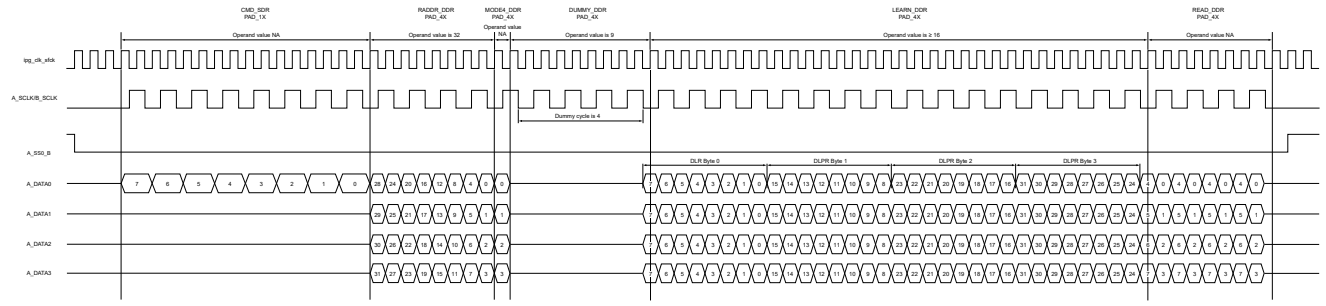


Figure 20. Flash memory access sequence example (data learning)

NOTE

- FlexSPI dummy instruction starts and ends at the rising edge of serial root clock.
- Device dummy cycle starts and ends at the falling edge of SCLK.
- The operand value of DUMMY_DDR instruction is odd because the total cycle number before DUMMY_DDR cycle is odd.

Figure 21 shows an example HyperBus device read transaction (single latency count) in individual mode.

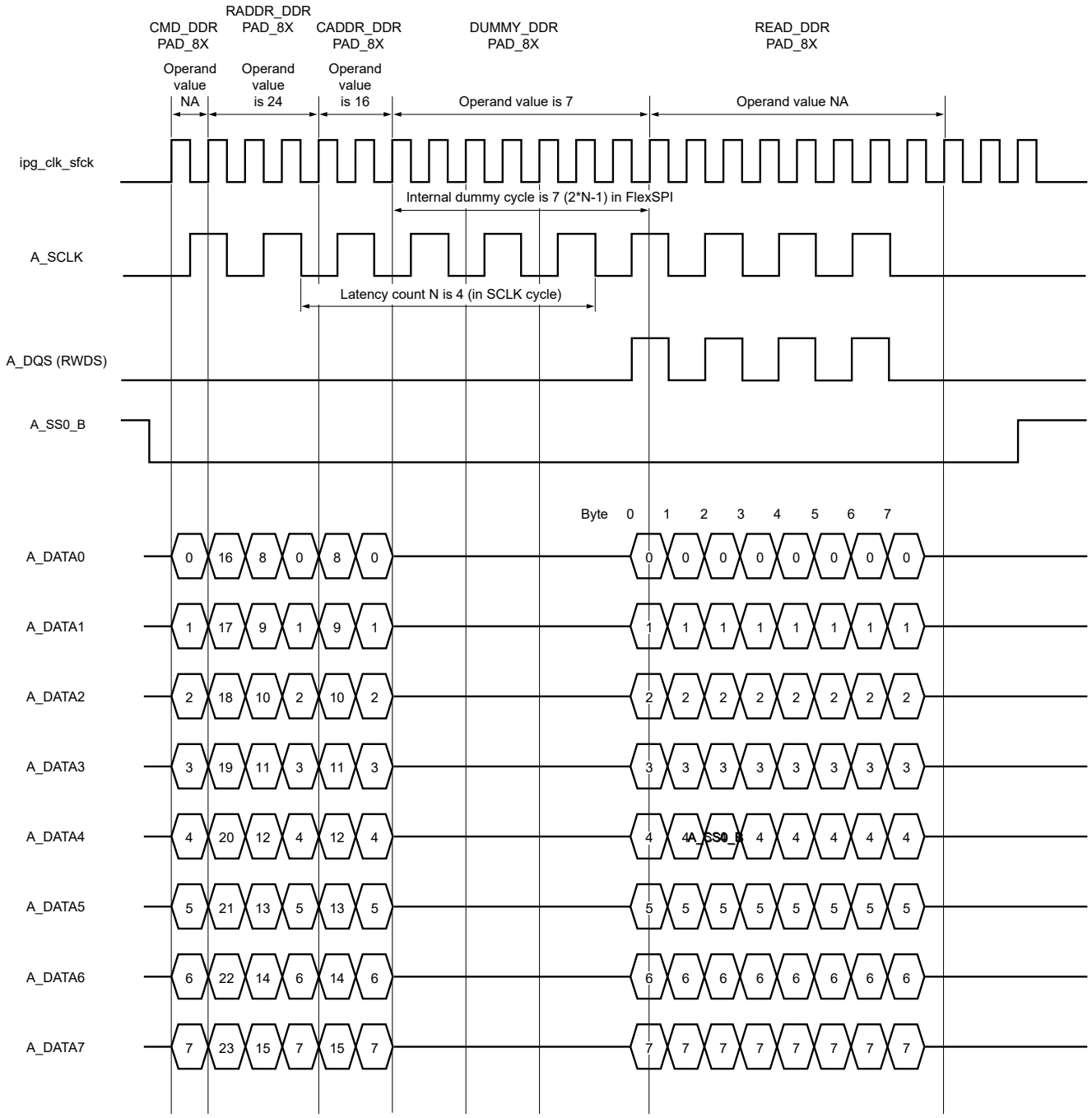


Figure 21. HyperBus device read transaction with single latency count

NOTE

FlexSPI continues to drive the clock until it has latched all of the read data it is expecting. On-chip delay for the data to reach FlexSPI can increase the number of SCK clock pulses.

Figure 22 shows an example HyperBus device read transaction (additional latency count) in individual mode.

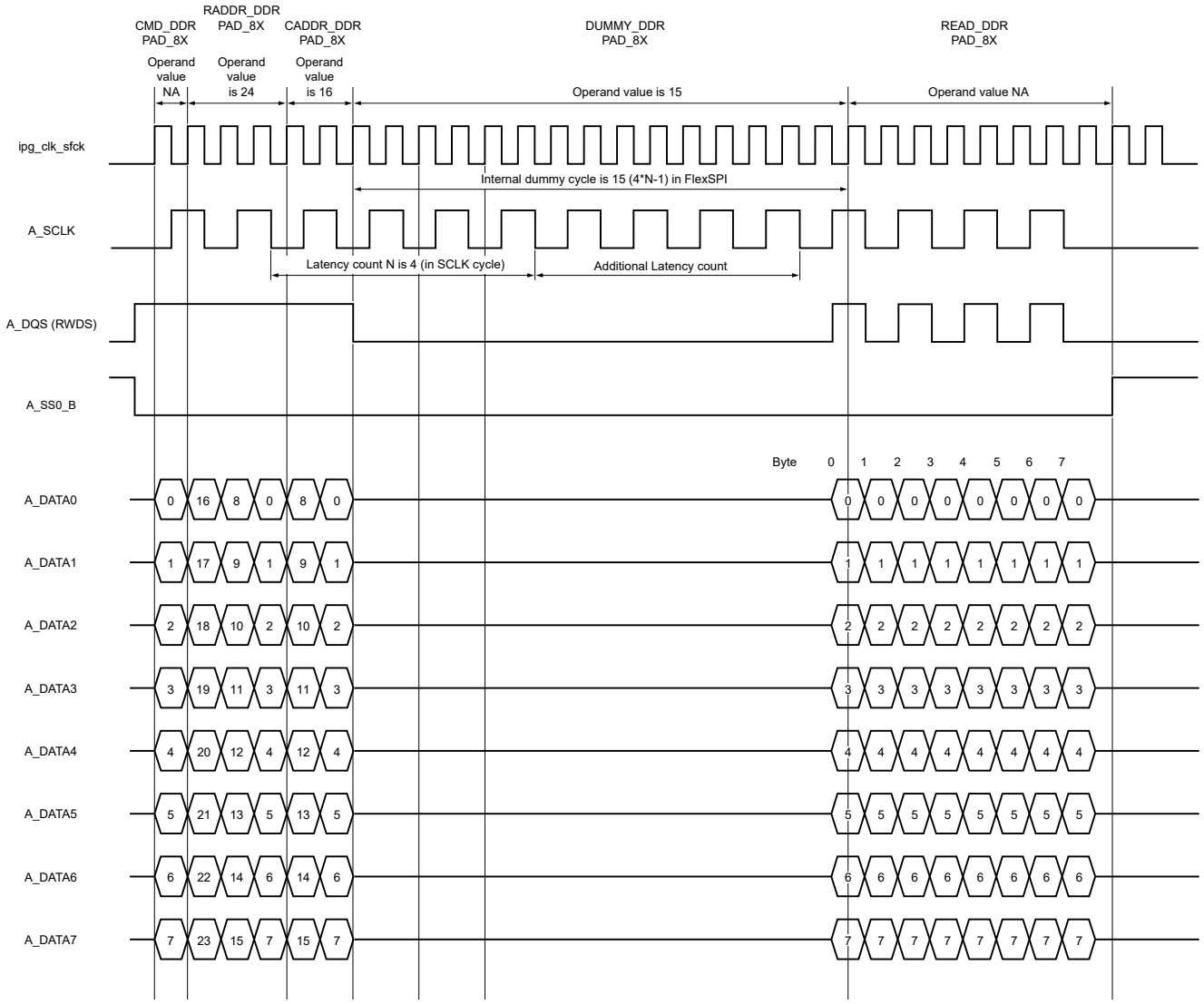


Figure 22. HyperBus device read transaction with additional latency count

Figure 23 shows an example HyperBus device write transaction (single latency count) in individual mode.

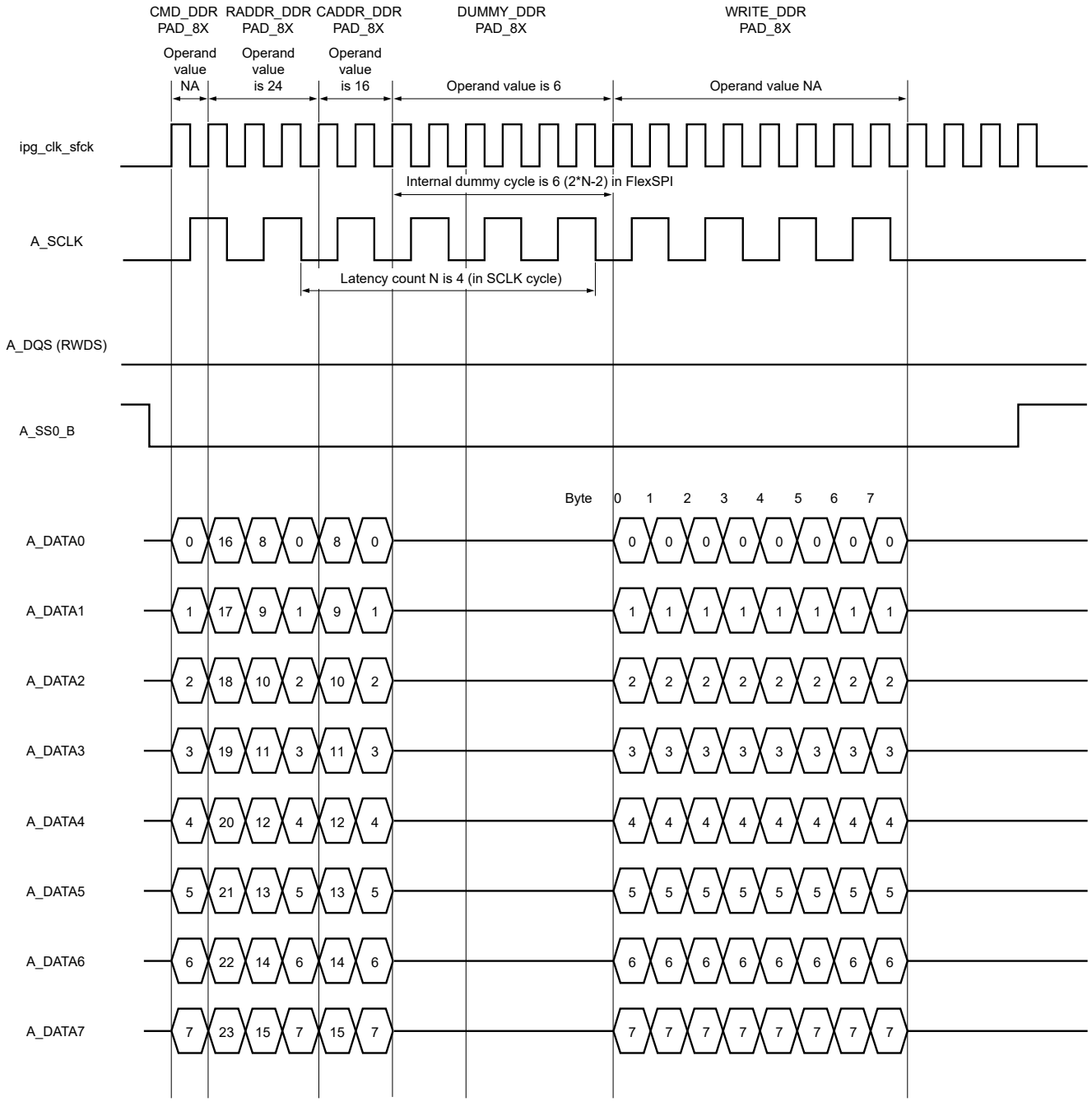


Figure 23. HyperBus device write transaction with single latency count

Figure 24 shows an example HyperBus device write transaction (additional latency count) in individual mode.

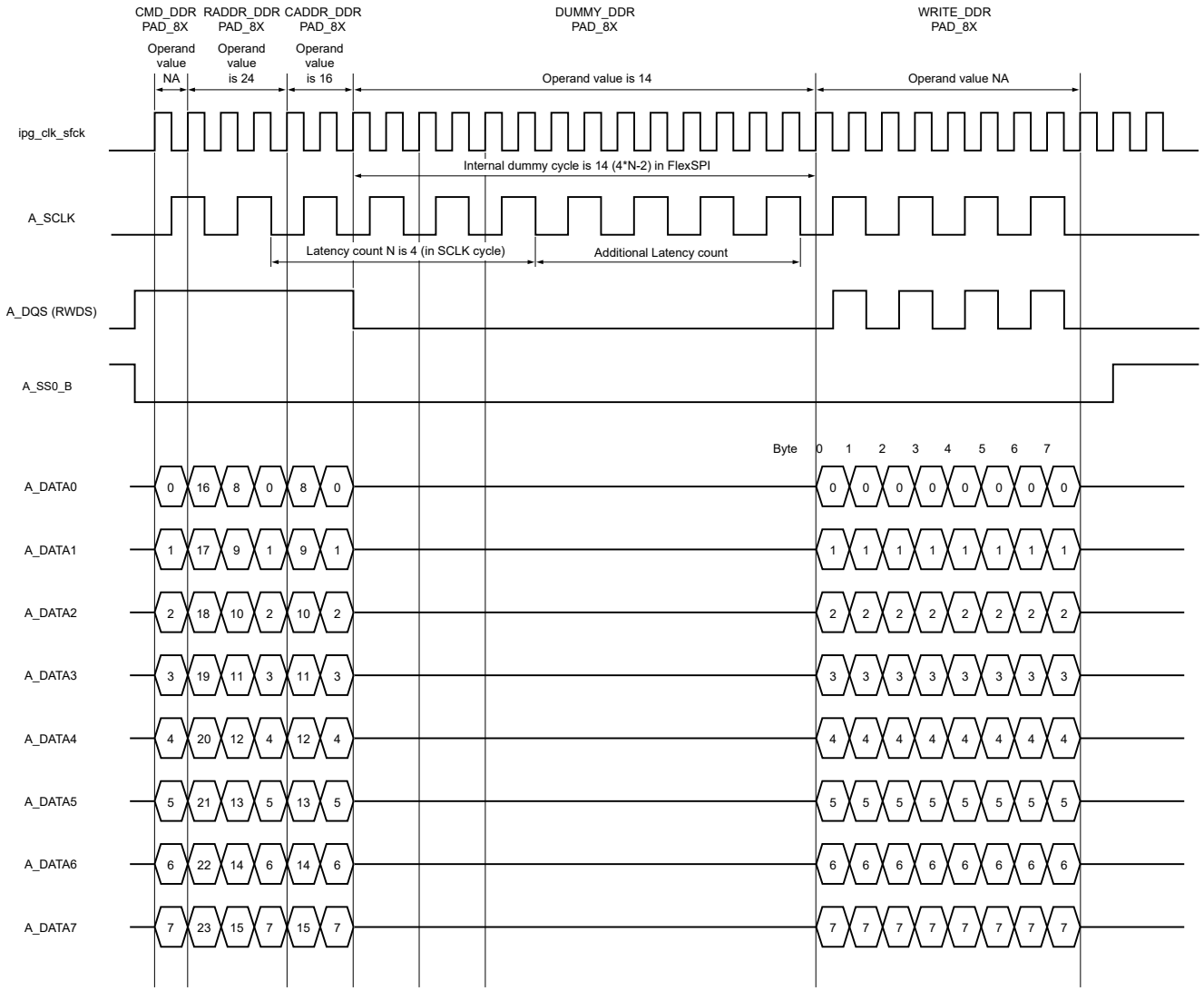


Figure 24. HyperBus device write transaction with additional latency count

10.3.7 Clocking

This section describes FlexSPI clocks and special clocking requirements.

Table 79. Clock usage

Clock name	Description	Comment
serial clock root (ipg_clk_sfck)	Root clock for Serial domain	FlexSPI core clock, used to generate the serial clock output. The clock is optionally divided using MCR0[SERCLKDIV] to obtain the SCLK.

Table continues on the next page...

Table 79. Clock usage (continued)

Clock name	Description	Comment
		Its frequency is same as SPI SCK frequency in SDR mode if MCR0[SERCLKDIV] = 0.
ahb clock (hclk)	AHB bus clock	hclk frequency is higher than 1/4 of serial root clock, so internal async FIFO is never full. This clock is used for internal logic in FlexSPI.
ipg clock (ipg_clk)	IPS bus clock	Register access clock. ipg_clk must be equal to hclk or an integer division of hclk. For example, ipg_clk = hclk/2.
SCLK	FlexSPI serial clock. Output clock on SCLK pin	Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the entire flash memory access sequence.
DQS_OUT	Dummy Read Strobe output	Same frequency as SCLK. Clock output toggles during READ and LEARN instructions. The DQS ensures that the data can be strobed correctly. It is only used for data strobe, instead of command or address strobe. The data includes useful read data and data learning patterns.
DQS_IN	Sample clock for receive data	Same frequency as SCLK. Sample clock comes from looped-back dummy read strobe, looped-back SCLK, or flash-memory-provided read strobe.

10.3.8 Interrupts

Table 80 describes all interrupts that FlexSPI generates.

Table 80. Flash memory address, row address, and column address

Interrupt	Enable	Condition
IP command done	INTEN[IPCMDDONEEN]	An IP command is finished.
IP command grant error	INTEN[IPCMDGEEN]	An IP command grant timeout occurs (bus not granted after MCR0[IPGRANTWAIT] × 1024 AHB clock cycles). See Overview of error categories, flags, and triggered sources.
AHB command grant error	INTEN[AHBCMDGEEN]	An AHB command grant timeout occurs (bus not granted after MCR0[AHBGRANTWAIT] × 1024 AHB

Table continues on the next page...

Table 80. Flash memory address, row address, and column address (continued)

		clock cycles). See Overview of error categories, flags, and triggered sources.
IP command error	INTEN[IPCMDERREN]	A command check error or a command execution error for an IP command occurs. See Overview of error categories, flags, and triggered sources.
AHB command error	INTEN[AHBCMDERREN]	A command check error or a command execution error for AHB command occurs. See Overview of error categories, flags, and triggered sources.
IP_RX_FIFO watermark available	INTEN[IPRXWAEN]	The fill level of IP_RX_FIFO reaches the watermark level (IPRXFCR[RXWMRK]).
IP_TX_FIFO watermark empty	INTEN[IPTXWEEN]	The empty level of IP_TX_FIFO reaches the watermark level (IPTXFCR[TXWMRK]).
Data learning failed	INTEN[DATALEARNFAILEN]	No valid sample clock phase found after LEARN instruction is executed. See Overview of error categories, flags, and triggered sources.
Sequence execution timeout	INTEN[SEQTIMEOUTEN]	A sequence execution time exceeds the timeout wait time (MCR1[SEQWAIT]). For example, the following flash memory read command sequence lasts about 800_0000h cycles (ipg_clk_sfck). If SEQWAIT is set to FFFFh, a sequence timeout interrupt is generated. • IP command triggers timeout. • Flash memory read data size is 100_0000h bytes. • Flash memory accesses in single mode and SDR mode.
AHB bus timeout	INTEN[AHBBUSTIMEOUTEN]	AHB bus response timeout occurs. For example, AHB read sequence is not configured properly in LUT (such as without a READ instruction). No data is read from an external device. As a result, FlexSPI cannot reach the read data in AHB_RX_Buffers for the AHB read command. See Overview of error categories, flags, and triggered sources.
Write command stops SCLK	INTEN[SCKSTOPBYWREN]	IP_TX_FIFO is empty while executing a write command sequence. FlexSPI stops SCLK output clock toggling and waits for write data filling.

Table continues on the next page...

Table 80. Flash memory address, row address, and column address (continued)

Read command stops SCLK	INTEN[SCKSTOPBYRDEN]	IP_RX_FIFO is full while executing a read command sequence. FlexSPI stops SCLK output clock toggling and waits to read the data from IP_RX_FIFO.
IP command security violation	INTEN[IPCMDSECUREVIOEN]	A non-secure controller requests an IP command sequence to an address outside the non-secure areas that IPSNSZSTARTn and IPSNSZENDn define.

10.3.9 Inline PRINCE encryption and decryption (IPED) CTR

IPED is a PRINCE-based encryption-decryption IP with a 64-bit block cipher and 128-bit key. FlexSPI supports automatic encryption on IP and AHB write commands, and automatic decryption on AHB read commands.

NOTE

The read or write access address sent to IPED.i_data is always 64 bits (eight bytes).

- For read operations, the start address and fetch size are always eight-byte-aligned. FlexSPI guarantees this condition.
- For write operations, software should manage any partial write that is not eight-byte-aligned. For example, before performing page erase and page program operations, the eight-byte boundary data should be read back.

Changing the register setting of IPEDCTRL changes the behavior of IPED.

To use IPED, apply these settings:

- AHBCR[READADDROPT] = 1
- AHBCR[PREFETCHEN] = 1 or AHBCR[READSZALIGN] = 1

FlexSPI supports a maximum of 7 address ranges for different 64-bit IPED.i_iv to encrypt or decrypt data.

The IPEDCTXnSTART and IPEDCTXnEND registers control the 7 ranges. There must be no overlap among these 7 ranges. Otherwise behavior of IP is undefined. Any single read or write request to flash memory must not cross one context boundary. For AHB read and write operations, hardware can automatically keep the boundary by writing 1 to AHBCR[ALIGNMENT]. Software must maintain the IP command write boundary. Any read or write request across context boundary leads to wrong data to or from the external device.

IPED context control 0 (IPEDCTXCTRL0) and IPED context control 1 (IPEDCTXCTRL1) control the write ability of IPEDCTXnSTART and IPEDCTXnEND. By default, IPEDCTXnSTART and IPEDCTXnEND cannot be modified after reset is enabled. Table 81 lists details.

Table 81. IPEDCTXnSTART and IPEDCTXnEND write ability

IPEDCTXCTRL0[CTXn_FREEZE0]	IPEDCTXCTRL1[CTXn_FREEZE1]	CTXn_FREEZE0 and CTXn_FREEZE1 writable	IPEDCTXnSTART and IPEDCTXnEND writable
10b	10b	yes	yes
01b (default)	10b (default)	yes	no
don't care	00b or 01b or 11b	no	no
00b or 11b	don't care	no	no

10.3.10 Flash memory access via IP command

Follow these steps to trigger a flash memory access via IP command.

1. If this command is a programing command, fill the IP transmit FIFO with programming data (for instance, programming flash data and flash memory status registers.)
2. Write the flash memory access start address to IPCR0[SFAR].
3. Write the read or program data size to IPCR1[IDATSZ]. Write the sequence index to IPCR1[ISEQID]. Write the sequence number to IPCR1[ISEQNUM].
4. Write 1 to IPCMD[TRG] to trigger a flash memory access command.
5. Wait for the INTR[IPCMDDONE] flag to set or for the IP command done interrupt to fire, indicating the command has completed on the FlexSPI interface.

NOTE

- The IP transmit FIFO can be filled before or after writing the IPCR0, IPCR1, and IPCMD registers. If the SFM command starts with the IP transmit FIFO empty, FlexSPI stops SCLK toggling to wait for transmit data ready automatically.
- The IPCMD register must be written after writing the IPCR0 and IPCR1 registers.
- One IP command can issue up to eight command sequences.
- You cannot issue another IP command before the previous IP command is finished. Performing this action results in unknown behavior.

If this command is a read command, all read data from flash memory is put into the IP receive FIFO. Software must read data from the IP receive FIFO via AHB bus or IP bus. When the IP receive FIFO is full and the flash device still contains data to read, FlexSPI stops SCLK toggling until the FIFO has free space. See SCLK stop.

A triggered serial flash command contains:

- A flash memory access start address. IPCR0[SFAR] determines this address.
- A flash chip select. The flash memory access address and flash memory size setting (FLSHxCR0[FLSHSZ]) determine the chip select.
- A flash command sequence index and sequence number. FlexSPI sequentially executes the sequences indexed from IPCR1[ISEQID] to (IPCR1[ISEQID] + IPCR1[ISEQNUM]) in LUT.
- Flash individual/parallel memory access mode:

  Determined by IPCR1[IPAREN].
- The flash memory read or program data size, in bytes.
  - If the value of IPCR1[IDATSZ] is nonzero, the data size is IPCR1[IDATSZ].
  - If the value of IPCR1[IDATSZ] is zero, the operand value in the read or write instruction determines the data size.

NOTE

- Software must ensure that the last sequence index never exceeds the LUT sequence number. That is, IPCR1[ISEQID] + IPCR1[ISEQNUM] must be less than 16.
- For sequence numbers greater than one, the data size is applied to every command sequence.
- If there is no write or read instruction in the command sequence, the data size is ignored.

The IP command request is sent to the arbitrator after software triggers it. It is not executed on the FlexSPI interface until the arbitrator grants it. See Command arbitration.

10.3.10.1 Reading data from IP receive FIFO

FlexSPI puts read data from the external device into the IP receive FIFO for IP commands.

The data stored in the IP receive FIFO can be read using IPS and register transactions:

- 400C8100h–400C817Ch (by IPS bus)

FlexSPI pushes read data into the IP receive FIFO in sets of 64 bits each time it receives 64 bits of data from an external device. When the number of read data bits is not 64-bit-aligned, FlexSPI pushes additional zero bits into the IP receive FIFO for the last push.

The processor or DMA can read the IP receive FIFO.

10.3.10.1.1 Reading via processor

To read data from the IP receive FIFO via processor, configure the items below.

- Write 0 to IPRXFCR[RXDMAEN].
- Write the watermark level to IPRXFCR[RXWMRK]. The watermark level is (IPRXFCR[RXWMRK] + 1) × 8 bytes.
- Write 1 to INTEN[IPRXWAEN] to enable the IP receive FIFO watermark available interrupt (optional).

The processor must poll INTR[IPRXWA] or wait for the IP receive FIFO watermark available interrupt before reading data from the FIFO. Waiting ensures that a watermark level amount of data is filled in the IP receive FIFO before reading.

After reading a watermark level amount of data from the FIFO, software must set the INTR[IPRXWA] flag to pop that data from the FIFO.

Figure 25 shows the reading flow from the IP receive FIFO via processor.

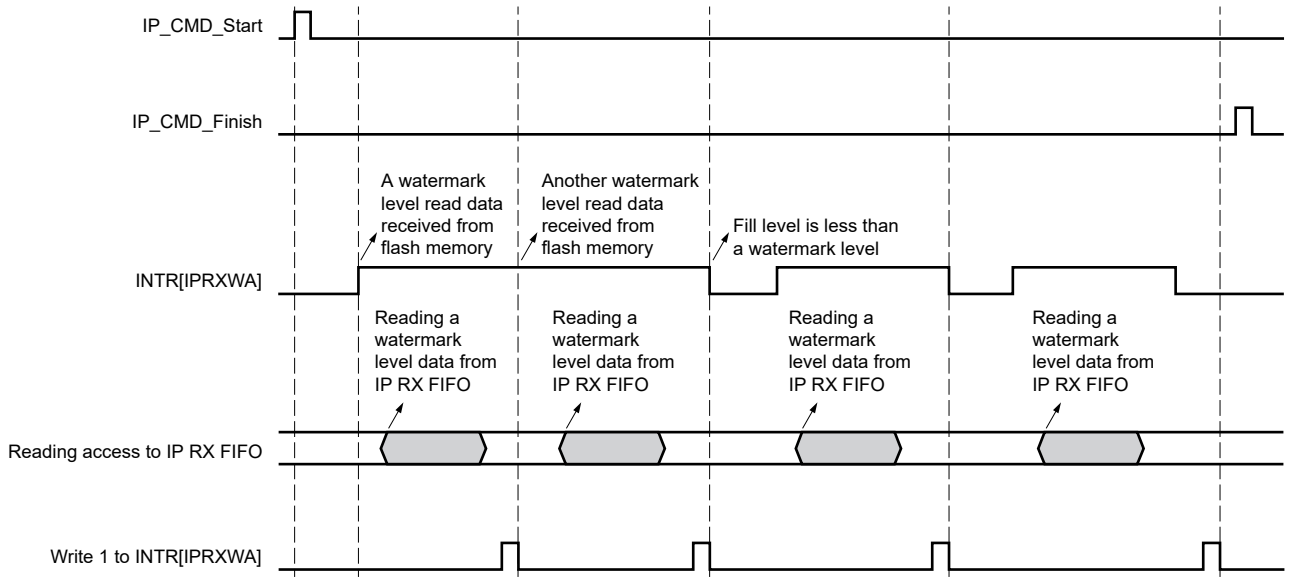


Figure 25. Reading IP receive FIFO via processor

NOTE

- The processor must read a watermark level of data from the IP receive FIFO before setting the INTR[IPRXWA] flag.
- The total flash read or program data size is not required to be a multiple of the watermark level. When the reading data size from the FIFO is less than a watermark level for the last time, software polls IPRXFSTS[FILL], not INTR[IPRXWA]. After all data from the FIFO are copied and all command sequences to flash memory are finished (INTR[IPCMDDONE] = 1), software sets IPRXFCR[CLRIPRXF] to clear the FIFO. If the FIFO is not cleared, the reading data will be incorrect for the next reading command to flash memory.
- Setting the INTR[IPRXWA] flag pops the IP receive FIFO data. Each read access to the FIFO does not pop the data.

10.3.10.1.2 Reading via DMA

To read data from the IP receive FIFO via DMA, configure the items below.

- Write 1 to IPRXFCR[RXDMAEN].
- Write the watermark level to IPRXFCR[RXWMRK]. The watermark level is (IPRXFCR[RXWMRK] + 1) × 8 bytes.
- Configure the DMA transfer minor loop size to match the watermark level.

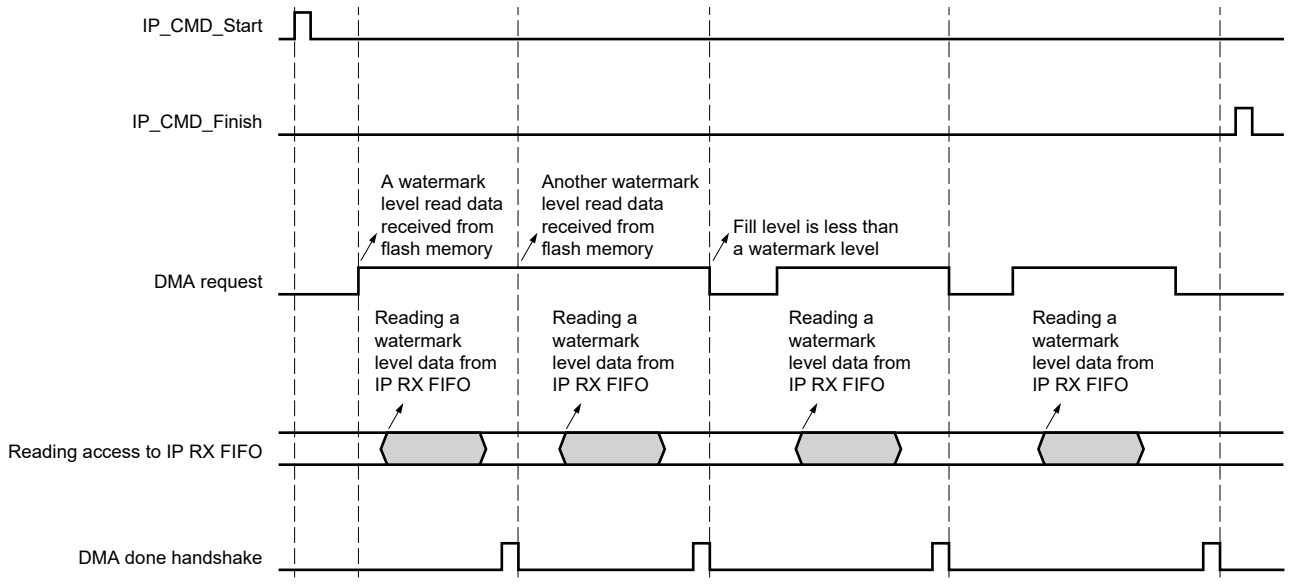


Figure 26. Reading from IP receive FIFO via DMA

NOTE

- When the fill level of the FIFO meets or exceeds the watermark level, a DMA request is generated. This request is level-valid, not pulse-valid.
- DMA reads a watermark level amount of data from the IP receive FIFO each time. (Configure the minor loop size to match the watermark level).
- DMA must return a done handshake (pulse valid) to FlexSPI each time it finishes reading a watermark level amount of data.
- The DMA done handshake (not each read access) pops the IP receive FIFO data.
- The total read and write data size (major loop size) must be a multiple of the watermark level. The DMA does not know when the data is ready for the last reading. When the data size is not a multiple of the watermark level, it is too complex for the DMA driver to poll IPRXFSTS[FILL].

10.3.10.2 Writing data to IP transmit FIFO

The programming data is put into the IP transmit FIFO, then FlexSPI transmits it to the flash memory. IPS can write it:

- 400C_8180h–400C_81FCh (via IP bus)

When FlexSPI fetches data to transmit, 64 bits of data from the IP transmit FIFO are popped. If the programming data size is not a multiple of 64 bits, the number of last popped valid bits is less than 64 bits. No error occurs; these invalid bits are not transmitted to the flash memory.

The processor or DMA can fill the IP transmit FIFO with programming data.

10.3.10.2.1 Writing via processor

To write data into the IP transmit FIFO via processor, configure the items below.

- Write 0 to IPTXFCR[TXDMAEN].
- Write the watermark level to IPTXFCR[TXWMRK]. The watermark level is (IPTXFCR[TXWMRK] + 1) × 8 bytes.
- Write 1 to INTEN[IPTXWEEN] to enable the IP transmit FIFO watermark available interrupt (optional).

The processor must poll INTR[IPTXWE] or wait for the IP transmit FIFO watermark empty interrupt before writing data to the FIFO. Waiting ensures that a watermark level amount of data is available in the IP transmit FIFO before writing.

After writing a watermark level amount of data to the FIFO, software must set the INTR[IPTXWE] flag to push that data into the FIFO. (The write pointer is incremented.)

NOTE

Setting the INTR[IPTXWE] flag pushes the IP transmit FIFO data. Each write access to the FIFO does not push the data.

Figure 27 shows the writing flow to the IP transmit FIFO via processor.

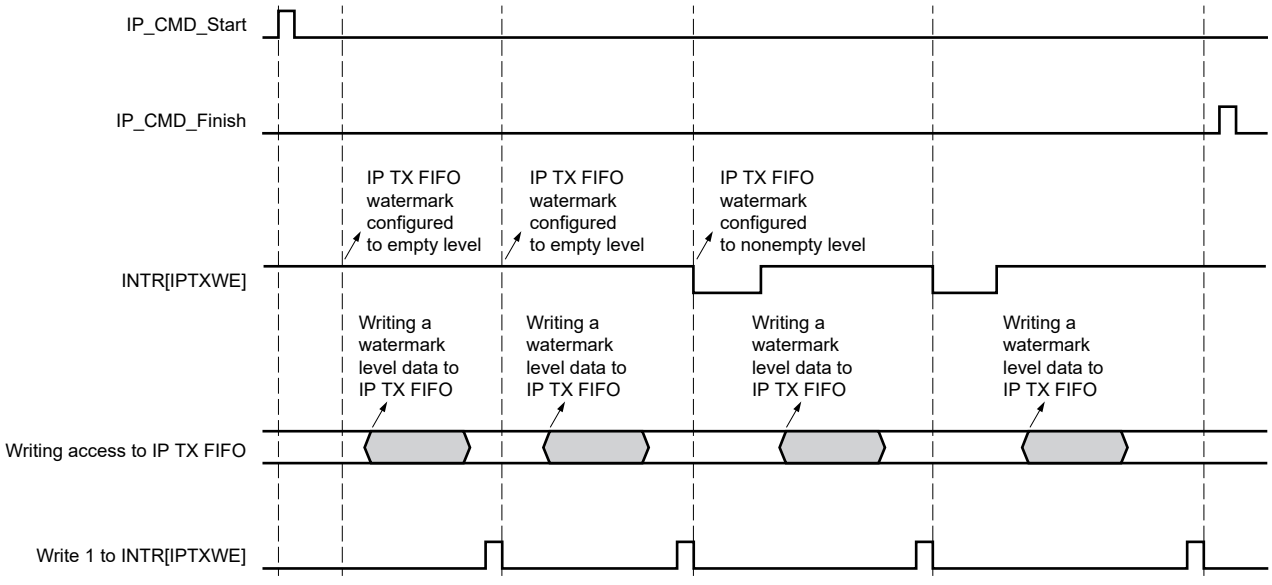


Figure 27. Writing to IP transmit FIFO via processor

NOTE

- The processor must write a watermark level amount of data into the IP transmit FIFO each time.
- The total flash write data size is not required to be a multiple of the watermark level. In this case, the writing data size to the FIFO is less than a watermark level for the last time. After all data to the FIFO are written and all command sequences to flash memory are finished (INTR[IPCMDDONE] = 1), the processor sets IPTXFCR[CLRIPTXF] to clear the FIFO. If the FIFO is not cleared, the programming data will be incorrect for the next programming command to flash memory.

10.3.10.2.2 Writing via DMA

To write data into the IP transmit FIFO via DMA, configure the items below.

- Write 1 to IPTXFCR[TXDMAEN].
- Write the watermark level to IPTXFCR[TXWMRK]. The watermark level is (IPTXFCR[TXWMRK] + 1) × 8 bytes.
- Configure the DMA transfer minor loop size to match the watermark level.

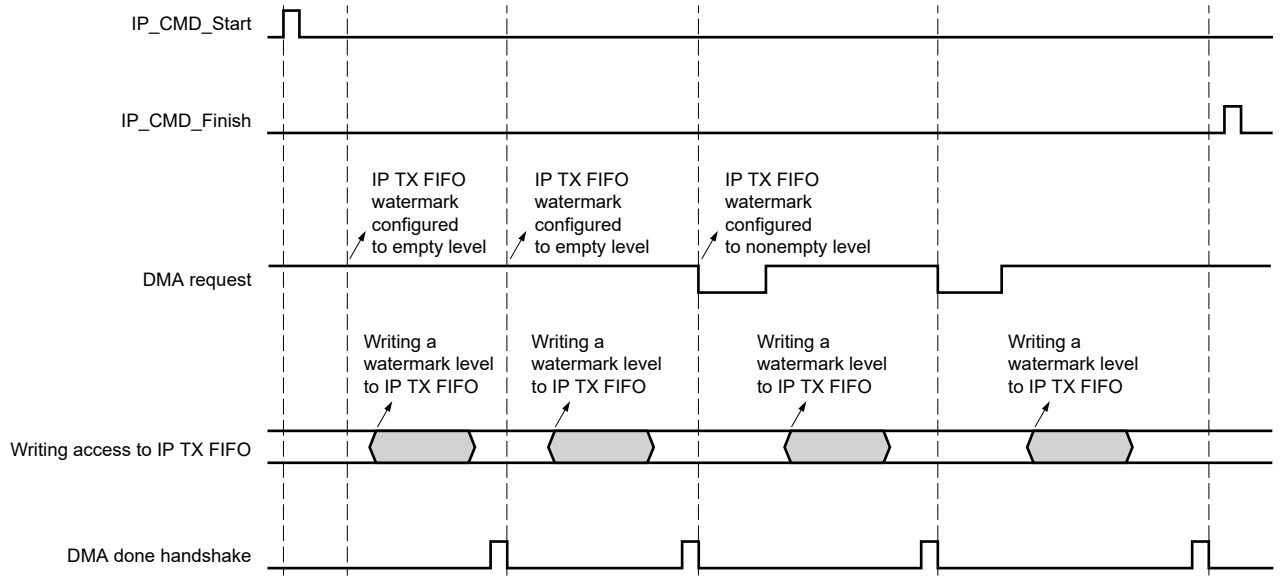


Figure 28. Writing into IP transfer FIFO via DMA

NOTE

- When there is empty space in the IP transmit FIFO greater than the watermark level, a DMA request is generated. This request is level-valid, not pulse-valid.
- DMA writes a watermark level amount of data into the FIFO each time. (Configure the minor loop size to match the watermark level.
- DMA returns a done handshake (pulse valid) to FlexSPI each time it finishes writing a watermark level amount of data.
- The DMA done handshake (not each write access) pushes the IP transmit FIFO data.
- The total program data size (major loop size) is not required to be a multiple of the watermark level. After writing all data into the IP transmit FIFO and all command sequences to flash memory are finished (INTR[IPCMDDONE] = 1), write 1 to IPTXFCR[CLRIPTXF] to clear the FIFO. Failure to clear the FIFO results in incorrect programming data for the next programming command to flash memory.

10.3.11 Flash memory access via AHB command

Flash memory can be accessed via AHB bus directly on the AHB address space. See chip-specific section for details about the AHB address space. This address space is mapped to serial flash memory in FlexSPI. AHB bus accesses to this address space trigger flash memory access command sequences as needed.

For AHB read access to serial flash memory, FlexSPI fetches data from flash memory into the AHB receive buffer, then returns the data on the AHB bus. For AHB write access to serial flash memory, FlexSPI buffers AHB bus write data into the AHB transmit buffer, then transmits the data to serial flash memory.

No software configuration or polling is needed for AHB commands, except for FlexSPI initialization. As with a normal AHB target, the AHB controller accesses the external flash device transparently.

AHB commands are normally used to access serial flash memory space. IP commands must be used to access the control and status registers or other spaces, such as a one-time programmable (OTP) space in an external flash device.

The AHB commands for read and write access are discussed in detail below.

10.3.11.1 AHB write access to flash memory

For AHB write access to flash memory, FlexSPI buffers the write data from the AHB bus into the internal AHB transmit buffer. FlexSPI then transmits the data to flash memory. FlexSPI only buffers write data for one AHB burst. Figure 29 shows the hardware operation in response to an AHB write access to flash memory.

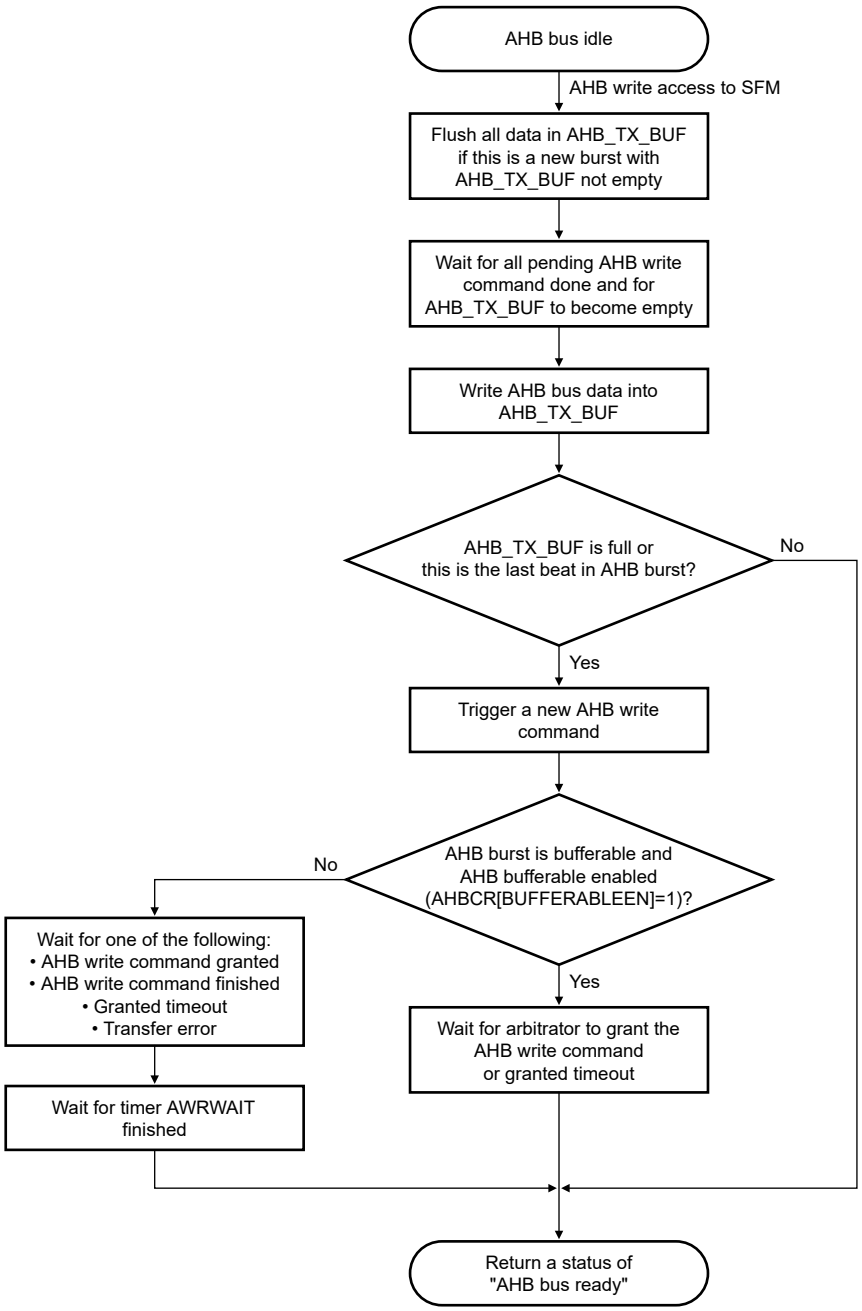


Figure 29. Hardware operation in response to AHB write access to flash memory

NOTE

1. If previous AHB burst is INCR write burst, AHB transmit buffer may not be empty when AHB bus is idle or new burst arrives. A new AHB write command is triggered to flush all data in AHB transmit buffer to external flash device. The new burst is held until this AHB write command finishes.
2. If the current access is bufferable, when the AHB write command is triggered, AHB ready is returned after the write command is granted. If the current access is non-bufferable, AHB ready is returned after the write command finishes.

FlexSPI triggers a new AHB write command in these cases:

- This beat is the last one in the current AHB burst, for any burst type except INCR.

- AHB transmit buffer is full after the current beat data is buffered.
- AHB bus becomes idle or a new burst arrives with the AHB transmit buffer not empty.

Table 82 describes the details of a triggered AHB write command.

Table 82. Triggered AHB write command

Item	Description
Flash memory access start address	Determined by AHB burst address. FlexSPI records the start address for the data in AHB transmit buffer, which is used as flash memory access start address.
Flash memory chip select	FlexSPI uses the settings of the flash memory access start address and flash memory size to determine the chip selection.
Flash memory command sequence index	Determined by FLSHxCR2[AWRSEQID].
Flash memory command sequence number	Determined by FLSHxCR2[AWRSEQNUM]. If AWRSEQNUM is not zero, multiple flash memory access command sequences are triggered every time for an AHB write command. The sequences indexed from AWRSEQID to (AWRSEQID + AWRSEQNUM) in LUT are executed sequentially.
Flash memory access mode (Individual/ Parallel)	Determined by AHBCR[APAREN].
Flash memory data size	Determined by byte number of buffered data in AHB transmit buffer.

The following examples indicate internal logic for AHB write access to flash memory, where AHB_TX_BUF is 64 bytes :

- AHB INCR 64-bit bufferable burst with address sequence 8h, 10h, 18h, 20h, ..., 50h (10 beats total).

  Two AHB write commands are triggered: the first command with flash memory start address 8h and data size 40h; the second command with flash memory start address 48h and data size 10h. See Figure 30.

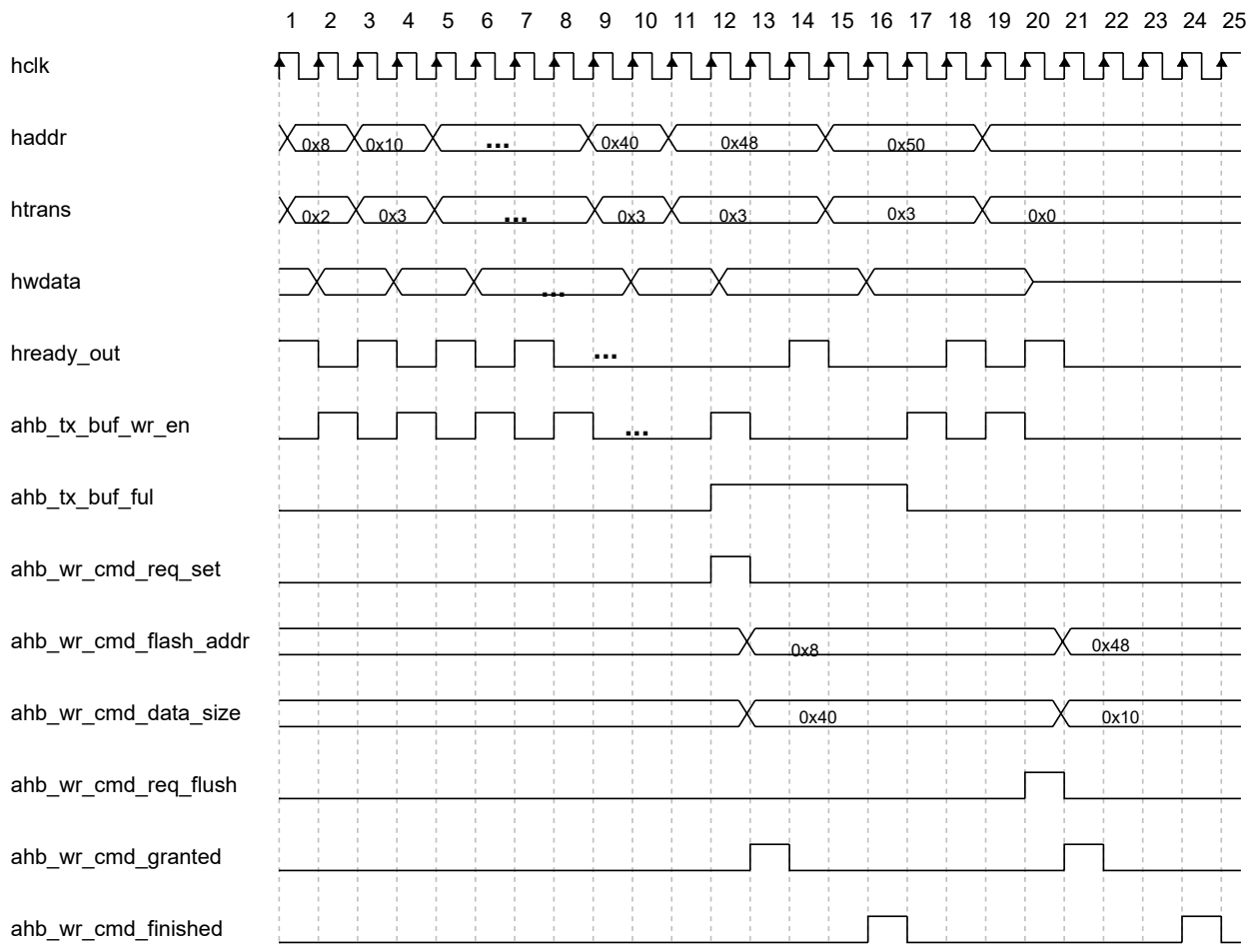


Figure 30. AHB write access (INCR, 64-bit, bufferable)

- AHB WRAP8 64-bit bufferable burst with address sequence 28h, 30h, 38h, 0h, 8h, 10h, 18h, 20h.

  One AHB write command is triggered with flash memory start address 0h and data size 40h. See Figure 31.

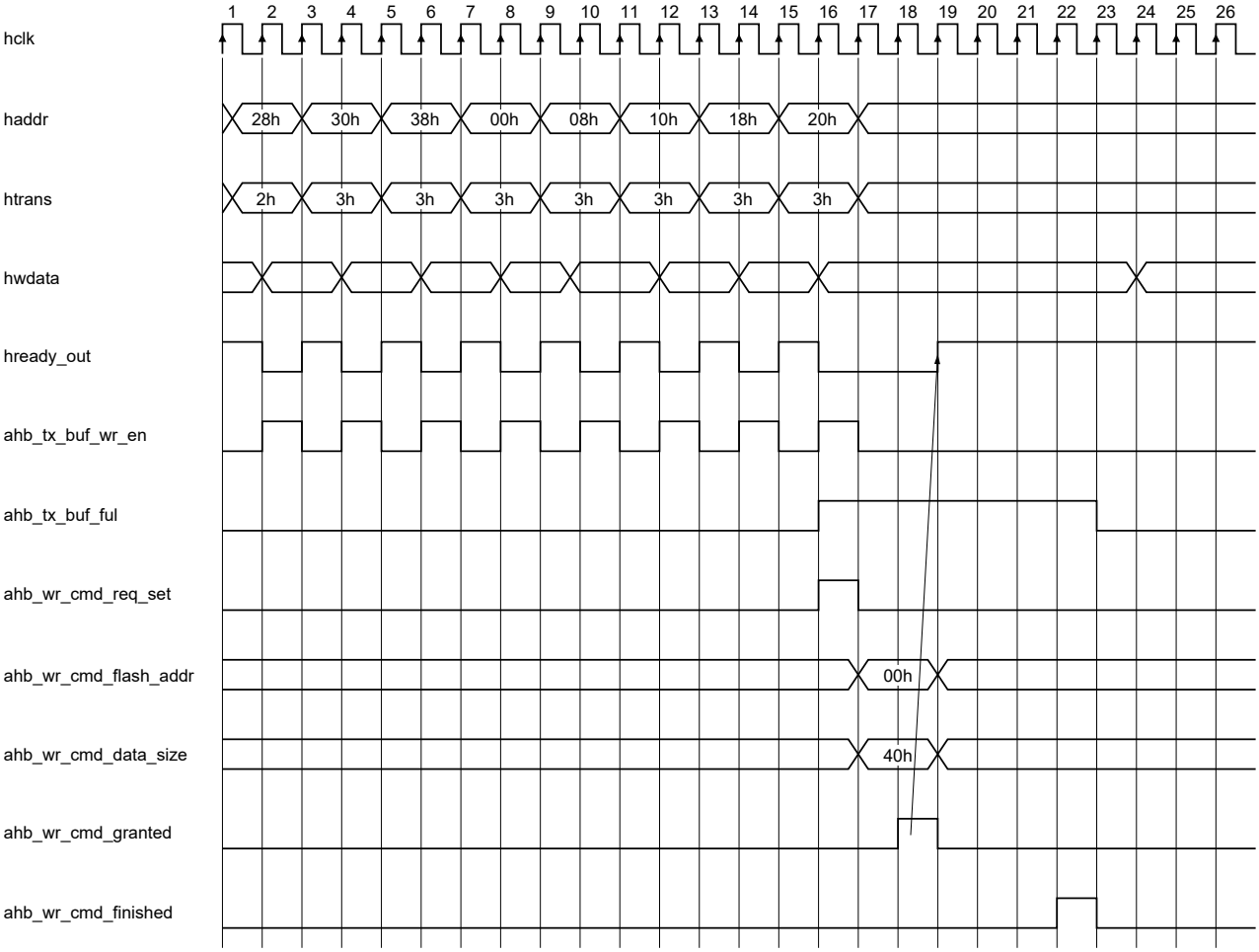


Figure 31. AHB write access (WRAP8, 64-bit, bufferable)

- AHB WRAP8 64-bit non-bufferable burst with address sequence 28h, 30h, 38h, 0h, 8h, 10h, 18h, 20h.

  One AHB write command is triggered with flash memory start address 0h and data size 40h.

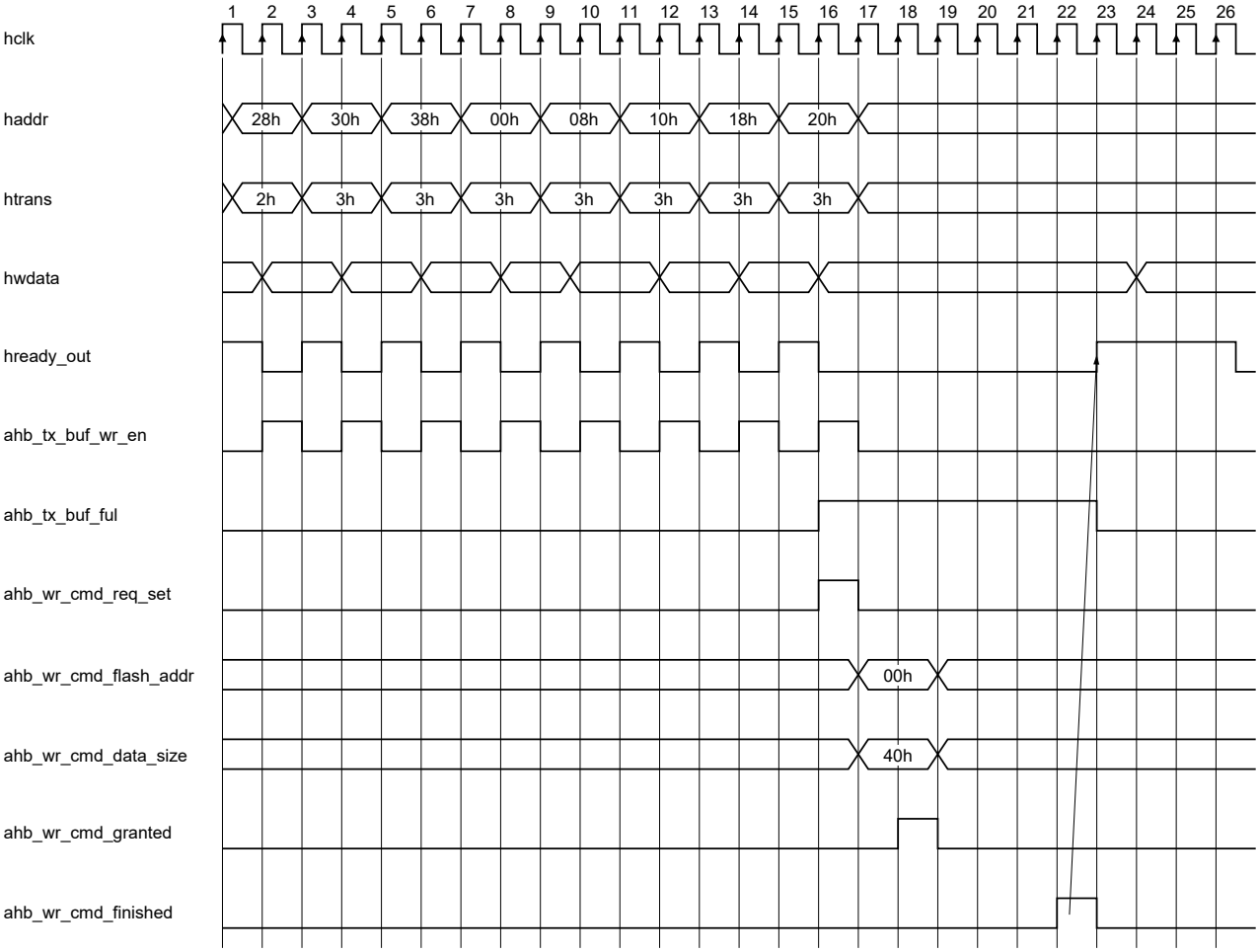


Figure 32. AHB write access (WRAP8, 64-bit, non-bufferable)

NOTE

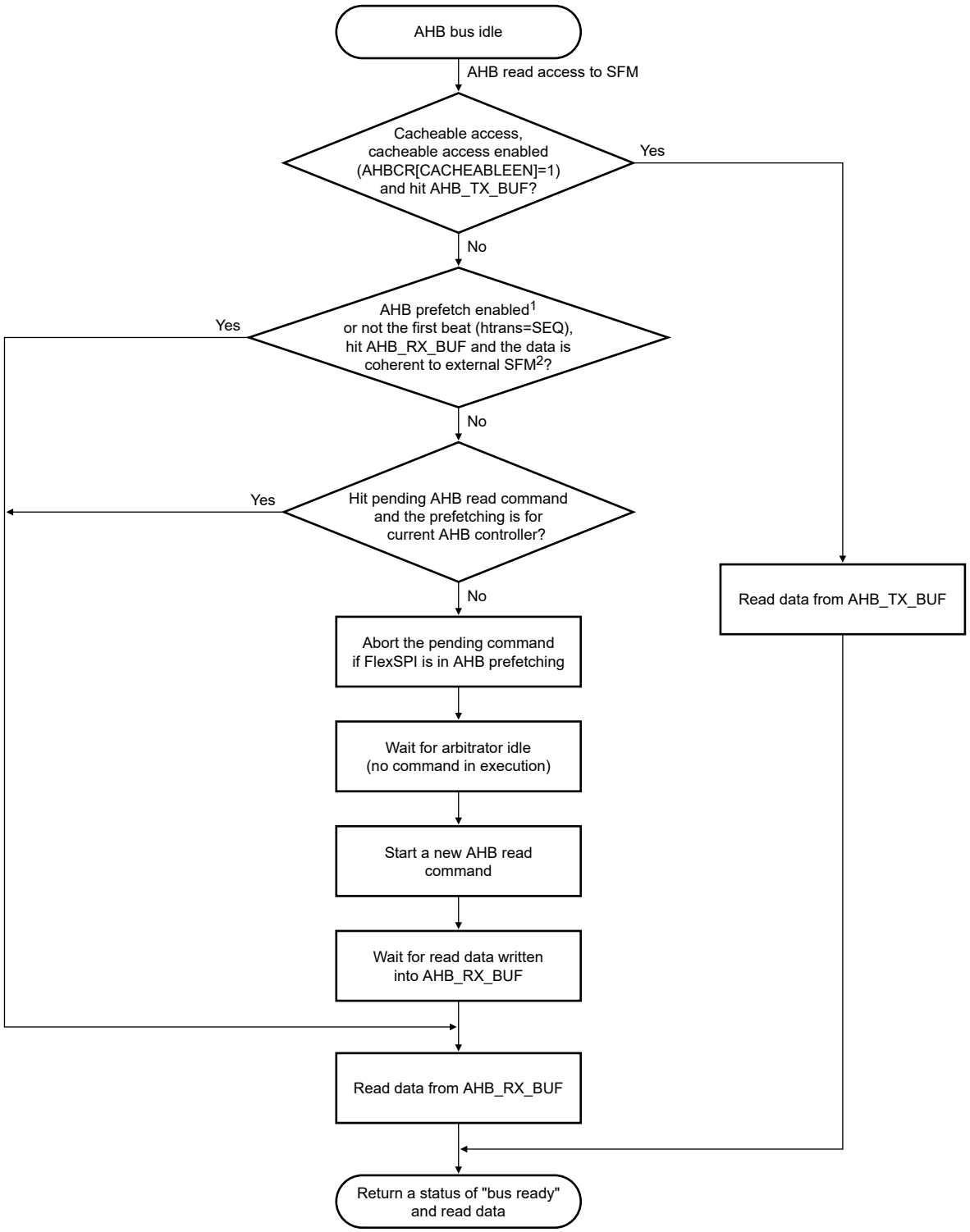
If the burst data size (in bytes) is greater than the AHB transmit buffer size, wrapper burst is not supported. For example, if AHB_TX_BUF is 64 bytes, AHB WRAP16 (16 × 64 bits = 128 bytes) write burst access is not supported.

10.3.11.2 AHB read access to flash memory

For AHB read access to flash memory, FlexSPI checks the burst access type and register setting to determine whether to access these locations:

- The AHB transmit buffer
- The AHB receive buffer
- A pending AHB read command

If all of these options are missed, FlexSPI triggers a new AHB read command to fetch data from flash memory. Figure 33 shows the hardware operation in response to AHB read access to flash memory.



Notes:
1) AHB prefetch is enabled when both AHBCR[PREFETCHEN]=1 and AHBRXBUFxCR0[PREFETCHEN]=1 (x is the AHB RX buffer ID for the current AHB controller).
2) The AHB RX buffer holds the data read from flash memory. This data may become incoherent if AHB writes to the same flash-memory address.

Figure 33. Hardware operation in response to AHB read access to flash memory

Table 83 describes the details of a triggered AHB read command.

Table 83. Triggered AHB read command

Item	Description
Flash memory access start address and data size	Determined by AHB address, burst type, and burst size. FlexSPI fetches read data from the start address for the current burst or beat. See Table 84 for details.
Flash memory chip select	FlexSPI uses the settings of the flash memory access start address and flash memory size to determine the chip select.
Flash memory command sequence index	Determined by FLSHxCR2[AWRSEQID].
Flash memory command sequence number	Determined by FLSHxCR2[AWRSEQNUM]. If AWRSEQNUM is not zero, multiple flash memory access command sequences are triggered every time for an AHB read command. The sequences indexed from AWRSEQID to (AWRSEQID + AWRSEQNUM) in LUT are executed sequentially.
Flash memory access mode (Individual/ Parallel)	Determined by AHBCR[APAREN].

NOTE

- FlexSPI automatically determines which ARDSEQNUM and ARDSEQID fields that the flash device uses as sequence ID for chip selection. See MCR2[SAMEDEVICEEN].
- FlexSPI determines which AHB receive buffer to use for the current AHB read access by controller ID. For details about the AHB receive buffer ID and AHB controller ID mapping, see AHB receive buffer management.
- You cannot allocate the size of the AHB receive buffer to be less than the AHB Burst size. This allocation results in unknown behavior.

Table 84. AHB read command flash memory start address and data size

Prefetch enable	Cross flash boundary	Burst type	Flash memory start address [28:0]	Data size
0	Never cross flash boundary, because AHB burst never crosses 1 KB boundary.	SINGLE, INCR4, INCR8, INCR16	hbeat_start_address	(hburst_end_address-hbeat_start_address)
		INCR	hbeat_start_address	byte size of current beat For INCR burst with prefetch disabled, each beat is handled same as SINGLE burst.
		WRAP4, WRAP8, WRAP16	hburst_start_address	(hburst_end_address-hburst_start_address)
1	No	INCR, SINGLE, INCR4, INCR8, INCR16	hbeat_start_address	ahb_rx_buf_sz
	No	WRAP4, WRAP8, WRAP16	hburst_start_address	ahb_rx_buf_sz

Table continues on the next page...

Table 84. AHB read command flash memory start address and data size (continued)

Prefetch enable	Cross flash boundary	Burst type	Flash memory start address [28:0]	Data size
	Yes	INCR, SINGLE, INCR4, INCR8, INCR16	hbeat_start_address	(flash_top_address-hbeat_start_address)
	Yes	WRAP4, WRAP8, WRAP16	hburst_start_address	(flash_top_address-hburst_start_address)

NOTE

- hbeat_start_address is HADDR input from the AHB controller for current beat.
- hburst_start_address (for example 00h) is the lowest address for current burst. hburst_end_address (for example 10h) is the highest address in current burst plus 1. For example, WRAP4 burst with HADDR 8h, Ch, 0h, 4h.
- ahb_rx_buf_sz is the buffer size in bytes of AHB receive buffer that the current AHB controller uses.
- flash_top_address is the top address of currently accessed flash memory.

10.3.11.3 AHB receive buffer management

There are 8 buffers (Buffer 0–7) in the AHB receive buffer. These buffers are transparent to AHB controllers. FlexSPI fetches flash memory data and returns it on the AHB bus automatically. No status register polling is needed for AHB read access to Serial Flash Memory (SFM).

The total size of AHB receive buffers is 1 KB. Use AHBRXBUF0CR0[BUFSZ]–AHBRXBUFCR7[BUFSZ] to configure the buffer size of each AHB receive buffer. The buffer size for Buffer 0 to Buffer 7 could be set to 0. If the buffer size is set to 0, FlexSPI ignores the related setting of AHBRXBUFxCR0[MSTRID]. Buffer 7 is used for all AHB controllers not assigned to Buffer 0–6. FlexSPI ignores the Buffer 7 size setting field (AHBRXBUF7CR0[BUFSZ]). Its buffer size is: (AHB receive buffer total size) - (the sum of the sizes of Buffer 0–6).

When an AHB read access to serial flash memory occurs, FlexSPI determines which buffer to use via this method:

1. If the controller ID equals AHBRXBUF0CR0[MSTRID] and AHBRXBUF0CR0[BUFSZ] is not zero, Buffer 0 is used.
2. If the controller ID equals AHBRXBUF1CR0[MSTRID] and AHBRXBUF1CR0[BUFSZ] is not zero, Buffer 1 is used.
3. If the controller ID equals AHBRXBUF2CR0[MSTRID] and AHBRXBUF2CR0[BUFSZ] is not zero, Buffer 2 is used.
4. If the controller ID equals AHBRXBUF3CR0[MSTRID] and AHBRXBUF3CR0[BUFSZ] is not zero, Buffer 3 is used.
5. If the controller ID equals AHBRXBUF4CR0[MSTRID] and AHBRXBUF4CR0[BUFSZ] is not zero, Buffer 4 is used.
6. If the controller ID equals AHBRXBUF5CR0[MSTRID] and AHBRXBUF5CR0[BUFSZ] is not zero, Buffer 5 is used.
7. If the controller ID equals AHBRXBUF6CR0[MSTRID] and AHBRXBUF6CR0[BUFSZ] is not zero, Buffer 6 is used.
8. If all the above cases are not met, Buffer 7 is used.

NOTE

- Software must ensure that the size of each buffer is not less than the maximum burst size of an AHB read access from the AHB controller using this buffer. Otherwise, the behavior is undefined. The minimum buffer size request depends on the chip implementation. For example, if the capability of the chip to send out a maximum burst is 64 bytes, the minimum AHB receive buffer is 64 bytes.
- Assigning multiple buffers to a single AHB controller is not supported, unless the prefetch buffer enhancement feature is being used (see Prefetch buffer enhancement).
- When AHB read prefetch is enabled (AHBCR[PREFETCHEN] = 1), the size of the buffer determines the prefetch data size. If it does not cross the flash boundary, FlexSPI fetches data from the external flash device with that buffer size.
- The priority setting of AHB controller (AHBRXBUFxCR0[PRIORITY] = 1) is used only for suspending control of AHB prefetching. See Command abort and suspend.

10.3.11.3.1 Prefetch buffer enhancement

If a given controller frequently accesses memory at multiple address areas (non-sequential accesses), then the prefetch buffer can become less effective. For example, if the CPU is executing task A at a given address, then switches to task B at a different address, then when the CPU switches to task B (FLEXSPI receives request for an access that does not hit in the prefetch buffer), then the prefetch buffer will be flushed and a new prefetch will start at the task B address. When the CPU switches back to task A (task A addresses now miss in the prefetch buffer, because the buffer holds task B addresses instead), then the prefetch buffer will be flushed again and start prefetching at the task A address.

If the switches between task A and task B happen frequently, then FLEXSPI can end up re-reading items it had previously stored in the prefetch buffer as the buffer ends up being flushed often due to the task switches.

To increase the efficiency of FLEXSPI in situations like those described in the example above, the prefetch buffer enhancement feature can be enabled. When prefetch buffer enhancement is enabled (AHBRXBUFnCR0[REGIONEN]=1), then multiple prefetch buffers can be assigned to the same controller with an address range (AHBBUFREGIONSTARTn/ENDn) used as an additional filter to determine which buffer is used.

FLEXSPI compares the controller ID along with corresponding address range of each buffer to select one buffer. Other buffers are not impacted. This way, the AHB controller can prefetch the data from multiple address ranges in each of the buffers.

Program sequence:

- Configure AHBBUFREGIONSTARTn and AHBBUFREGIONENDn with the desired address ranges. Note that the addresses here use the AHB address including the system base address, not the serial flash address with the base address removed
- END must be larger than START
- Set AHBRXBUFnCR0[REGIONEN] = 1
- Set AHBRXBUFnCR0[MSTRID] with the MSTRID corresponding to the controller that will own multiple buffers, for example 4'b0011

NOTE

- This function is only meaningful when prefetch is enabled.
- Only Buffer 0 - 3 support this enhancement when REGIONEN=1, other buffers work like before. Buffer 0 - 3 with REGIONEN=0 works like before.

Prefetch buffer enhancement example:

The core in the system uses MSTRID = 0. The core will frequently switch between task A at 0x8000_1000-0x8000_1FFF and task B at 0x8000_2000-0x8000_2FFF, so dedicated prefetch buffers will be used for these tasks. The core will access other addresses in the FLEXSPI region using a third prefetch buffer.

To achieve this configuration:

- AHBBUFREGIONSTART0 = 0x8000_1000 and AHBBUFREGIONEND0 = 0x8000_2000. Configure Buffer 0 to the address range used by task A.
- AHBBUFREGIONSTART1 = 0x8000_2000 and AHBBUFREGIONEND1 = 0x8000_3000. Configure Buffer 1 to the address range used by task B.
- AHBRXBUF0CR0 = 0xC000_0004. Buffer 0 is a 256 bytes buffer assigned to the core (MSTRID = 0) with the prefetch buffer enhancement feature enabled (REGIONEN = 1), so only task A addresses will use buffer 0.
- AHBRXBUF1CR0 = 0xC000_0004. Buffer 1 is a second 256 bytes buffer assigned to the core (MSTRID = 0) with the prefetch buffer enhancement feature enabled (REGIONEN = 1), so only task B addresses will use buffer 1.
- AHBRXBUF4CR0 = 0x8000_0004. Buffer 4 is a third 256 bytes buffer assigned to the core (MSTRID = 0). The prefetch buffer enhancement feature will not be used for this buffer (and is not available for buffer 4 anyway). Core accesses that do not hit in Buffer 0 or Buffer 1 will use Buffer 4.

10.3.12 Command arbitration

There are four flash memory access command sources:

1. AHB command, that AHB write access to serial flash memory space triggers
2. AHB command, that AHB read access to serial flash memory space triggers
3. IP command, that writing IPCMD[TRG] triggers
4. Suspended command, an AHB read prefetch sequence which is suspended

An AHB bus access never triggers a write command and a read command at the same time.

The AHB prefetch sequence is an AHB command sequence triggered via AHB read access when AHB prefetch is enabled. After all read data is fetched for the current AHB read burst, FlexSPI prefetches more data to reduce read latency for the next AHB read access. An AHB command for a read operation is never aborted while fetching read data for the current AHB read burst. However, a new IP command or AHB command request while prefetching data (not for the current read burst) can abort an AHB read command.

When the arbitrator is idle (STS0[ARBIDLE] = 1), the granted priority of these command sources is:

1. AHB command (read or write)
2. IP command
3. Suspended command

NOTE

When the arbitrator is idle and there is no AHB or IP command request, a suspended command is not granted immediately. The arbitrator waits MCR2[RESUMEWAIT] AHB clock cycle idle states before resuming the suspended command to avoid AHB prefetch sequence being resumed and suspended frequently.

If the arbitrator is busy executing AHB or IP commands (not suspended commands), no new command requests are granted. If the grant times out, an AHB or IP command granted error occurs.

If a new AHB or IP command request comes while arbitrator is executing AHB read prefetch sequence, AHB read prefetch sequence is aborted (but not immediately). Arbitrator grants the AHB or IP command request after AHB read prefetch sequence is aborted on the FlexSPI interface and saved all internal data pointers.

10.3.12.1 Command abort and suspend

Table 85 describes the command abort and suspend mechanism.

Table 85. Command abort and suspend

Action	Description
Aborting a command	As mentioned above, if a new AHB or IP command request arrives, an AHB read prefetch sequence can be aborted.
Suspending a command	When an AHB read prefetch sequence is aborted on the FlexSPI interface, FlexSPI saves this suspended sequence in the cases listed below. FlexSPI resumes this sequence after the arbitrator has been idle for enough time. • There is no valid suspended command (AHBSPNDSTS[ACTIVE] = 0). This condition can occur when no command is suspended or when the suspended command is resumed. • The priority of aborted AHB read prefetch sequence is higher than the active suspend sequence. **NOTE** FlexSPI ignores and never resumes the original suspended sequence.
Activating a suspended command	The suspended command (internal status) becomes active when an AHB prefetch command is aborted and suspended. It becomes inactive in the following cases: • The arbitrator resumes the suspended command. • The AHB controller triggers a new AHB read command request using the same AHB receive buffer (buffer ID). **NOTE** MCR0[SWRESET] can be used to clear any suspended command. The saved suspended command is dropped and no resuming occurs. The suspended command must be cleared after the flash page program or erase. Otherwise, another command may lead to a period in which the flash memory cannot support a read command.

Figure 34 shows an example of the flow of command abort, suspend, or resume.

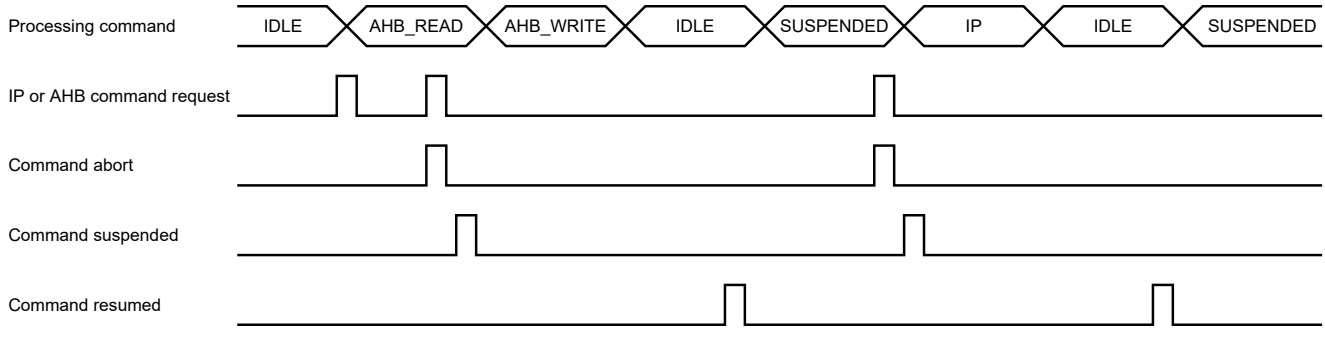


Figure 34. Command instruction execution on FlexSPI interface

NOTE

- The AHB prefetch sequence is aborted, suspended, or resumed twice in Figure 34. A new AHB write command request aborts the sequence the first time, and a new IP command request aborts the sequence the second time.
- The suspended sequence is resumed after a wait time of idle state. If there are frequent IP or AHB command requests and the wait time is not set reasonably, the suspend and resume activities may be triggered frequently.

10.3.13 SCLK stop

FlexSPI stops SCLK output toggling:

- When programming data is not ready for a programming command sequence.
- When an internal FIFO has no space to receive data for reading command sequence.

Certain devices may not support the stopping of SCLK during a command sequence (chip select is valid). SCLK stopping can be avoided via the methods described below.

- For flash memory reading that an IP command triggers:
  - Never trigger a read command with data size larger than IP receive FIFO size.
  - Internal asynchronous FIFO for flash memory reading should never be full.

    FlexSPI pops 64 bits of data per AHB clock cycle from this FIFO, and receives data from FlexSPI interface on the serial root clock. The flash memory access mode (Single, Dual, Quad, or Octal mode and Individual or Parallel mode) determines the receiving speed. For example, in Octal mode and Parallel mode, FlexSPI receives 16 bits per serial root clock cycle. If the AHB clock frequency is faster than 1/4 of the serial root clock, this asynchronous FIFO is never full.
- For flash memory programming that an IP command triggers:
  - Never trigger a program command with data size larger than the IP transmit FIFO size. Write all programming data into the IP transmit FIFO before triggering the IP command.
  - The internal asynchronous FIFO for flash memory programming must never be empty.

    FlexSPI fetches 64 bits of programming data per AHB clock cycle into this FIFO, and transmits data to the FlexSPI interface on the serial root clock. The flash memory access mode (Single, Dual, Quad, or Octal mode and Individual or Parallel mode) determines the transmitting speed. For example, in Octal mode and Parallel mode, FlexSPI transmits 16 bits per serial root clock cycle. If the AHB clock frequency is faster than 1/4 of the serial root clock, this asynchronous FIFO is never empty.
- For flash memory reading or programming that an AHB command triggers:
  - The internal asynchronous FIFO for flash memory reading and programming must never be full or empty. The frequency ratio limitation is same as it is for flash memory reading or flash memory programming that an IP command triggers.

NOTE

- FlexSPI never triggers an AHB read command with data size larger than the internal AHB receive buffer size.
- FlexSPI never triggers an AHB program command with data size larger than the internal AHB transmit buffer size.
- All programming data is buffered into the AHB transmit buffer before triggering an AHB program command in FlexSPI.

10.3.14 FlexSPI output timing

10.3.14.1 Output timing between data and SCLK

This section describes the output timing relationship of data (on A_DATA and B_DATA) and SCLK. There are three cases.

- **SDR instruction in SDR sequence**

  An SDR sequence contains only SDR instructions. In this case, all data bits last one serial root clock cycle on the FlexSPI interface. Figure 35 shows the relationship between the serial root clock, data, and SCLK:

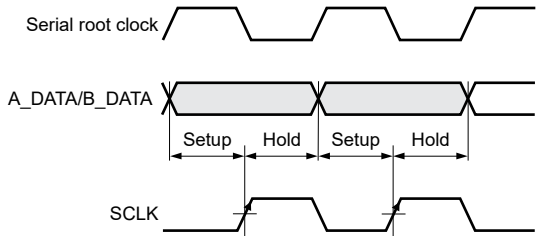


Figure 35. SDR instruction in SDR sequence

- **SDR instructions in a DDR sequence**

  A DDR sequence is a flash memory access command sequence that contains DDR instructions other than DUMMY_DDR. It may contain SDR instructions. If there are SDR instructions in a DDR sequence, all data bits last for two serial root clock cycles on the FlexSPI interface. Figure 36 shows the relationships between the serial root clock, data, and SCLK.

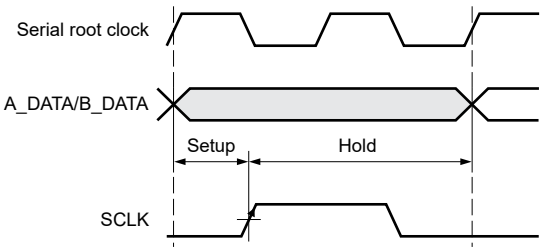


Figure 36. SDR instruction in DDR sequence

- **DDR instructions in a DDR sequence**

  For DDR instructions in a DDR sequence, all data bits last one serial root clock cycle on the FlexSPI interface. Figure 37 shows the relationships between the serial root clock, data, and SCLK.

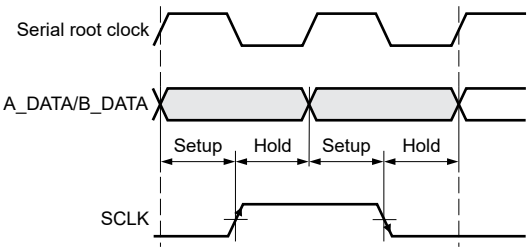


Figure 37. DDR instruction in DDR sequence

10.3.14.2 Output timing between chip select and SCLK

This section describes the output timing relationship between chip select (on A_SS0_B , A_SS1_B, B_SS0_B, and B_SS1_B) and SCLK. The timing relationship for the SDR sequence differs a little from the timing relationship for the DDR sequence.

- **Chip select timing in SDR sequence**

For an SDR sequence, the delay from chip select assertion to the SCLK rising edge is (FLSHxCR1[TCSS] + 0.5) cycles of the serial root clock. The delay from SCLK falling edge to chip select deassertion is FLSHxCR1[TCSH] cycles of serial root clock. Figure 38 shows the timing relationship between the chip select and SCLK.

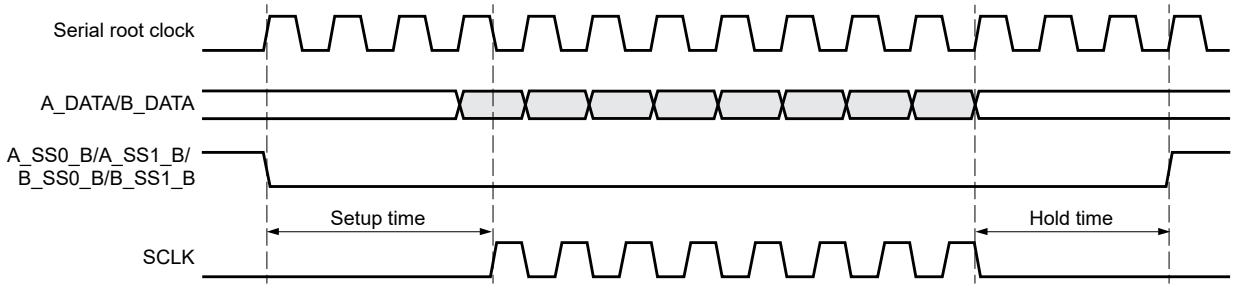


Figure 38. Chip select output timing for SDR sequence

- **Chip select timing in DDR sequence**

  For a DDR sequence, the delay from chip select assertion to the SCLK rising edge is **(FLSHxCR1[TCSS] + 0.5)** cycles of the serial root clock. The delay from the SCLK falling edge to chip select deassertion is **(FLSHxCR1[TCSH] + 0.5)** cycles of serial root clock.

  NOTE

  When AHB receive prefetch is enabled, and prefetch can be aborted, configure TCSH to 1 or greater. Configuring TCSH in this way guarantees a positive chip select hold time after the SCLK falling edge.

  Figure 39 shows the timing relationship between chip select and SCLK.

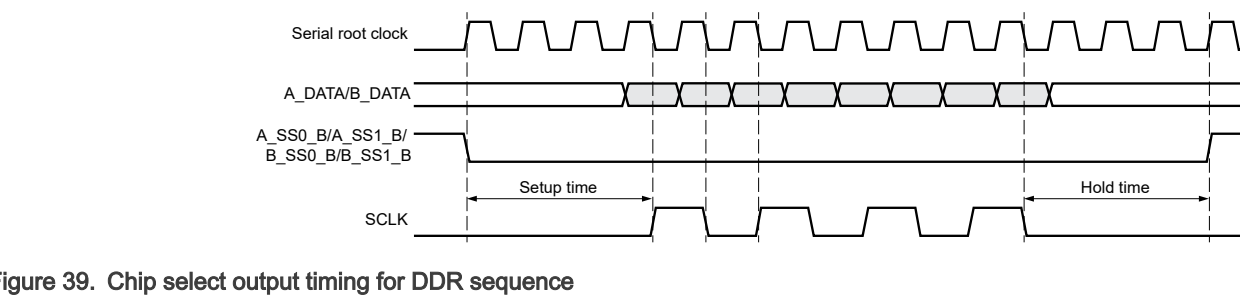


Figure 39. Chip select output timing for DDR sequence

For devices such as an FPGA device, a minimum chip select negation time (time between two chip select assertions) may be required. If the value of FLSHxCR1[CSINTERVAL] is nonzero, FlexSPI ensures a delay between chip select valid assertions. The delay time is CSINTERVAL × 256 cycles of serial root clock for an SDR or DDR sequence. If the external device has no limitation, configure the value of this field to zero. Figure 40 shows the timing of the chip select interval.

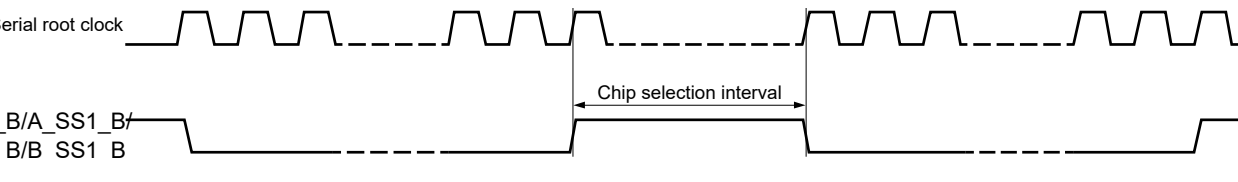


Figure 40. Chip select valid interval

10.3.15 FlexSPI input timing

This section describes the input timing of FlexSPI.

10.3.15.1 Receiving block

This section describes the receiving block in FlexSPI.

FlexSPI decodes instruction code to determine SDR mode or DDR mode. It also determines whether both the rising and falling edges are used for flash data sampling. FlexSPI samples the data line for learn instructions and read instructions.

MCR0[RXCLKSRC] selects the clock source from these four sources for the sampling clock:

- Internal dummy read strobe and looped back internally
- Internal dummy read strobe and looped back from DQS pad
- SCK output and looped back from SCK pad
- Flash-memory-provided read strobe

For details about DLL configuration, see DLL configuration for sampling.

Figure 41 shows the sampling block in FlexSPI.

FlexSPI uses both rising and falling edges to sample data from flash_A and flash_B. Two muxes with four clock source inputs generate the signals ipp_ind_io_fa, ipp_ind_io_fb. See Receive clock source features. ipp_ind_io_fa_int and ipp_ind_io_fb_int are reversed from ipp_ind_io_fa and ipp_ind_io_fb. Both sampling flip-flops are used when FlexSPI is in DDR mode. Only the falling edge sampling flip-flop is used in SDR mode.

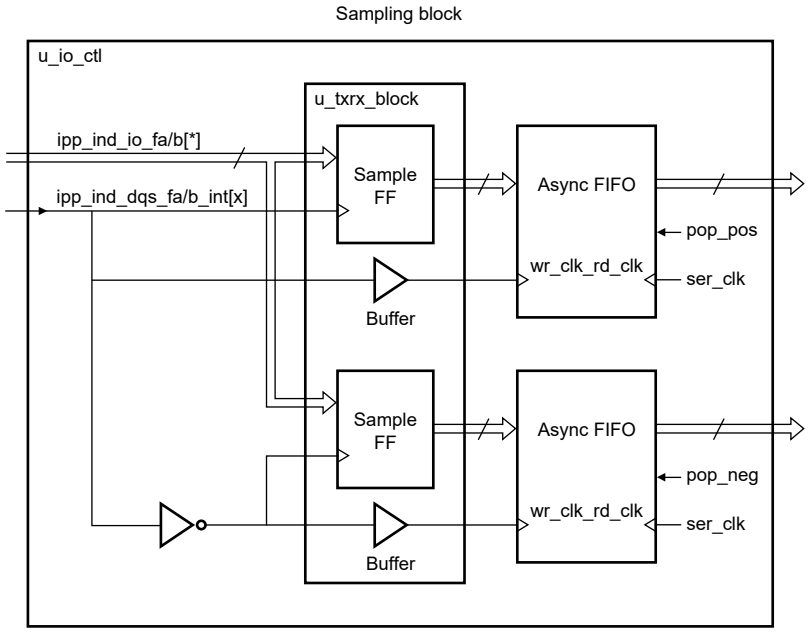


Figure 41. Sampling block in FlexSPI

Figure 42 is the receiving block diagram for data learning.

There are 16 clock phases for sampling data. An internal delay chain delays the clock phases.

See Data learning.

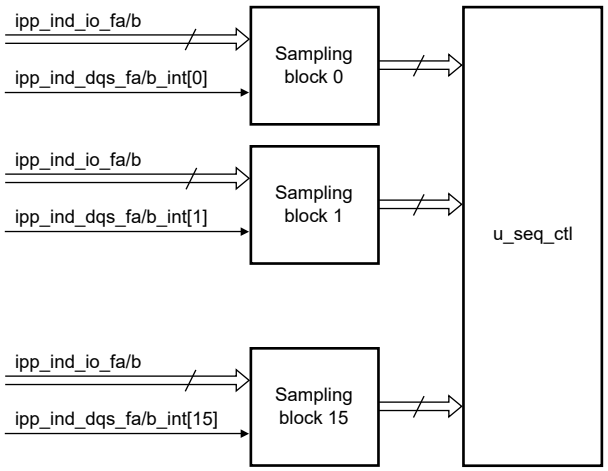


Figure 42. Sampling blocks for data learning

FlexSPI does not support read operations with a zero-turnaround cycle for direction switching on I/O pins. This read operation often occurs in QuadSPI configuration register reads. In Figure 43 and Figure 44, SIO[3:0] is an output from FlexSPI in the CMD phase, and is an input to FlexSPI in the DATA phase. No time remains for FlexSPI to switch the direction. This kind of read is not supported. FlexSPI and flash memory drive the I/O pins at the same time, which may cause a problem. To avoid this behavior on I/O pins, using SPI mode or reading with dummy cycles is required.

NOTE

The cycle numbers are not accurate in the examples below. See the device spec for accurate cycle numbers.

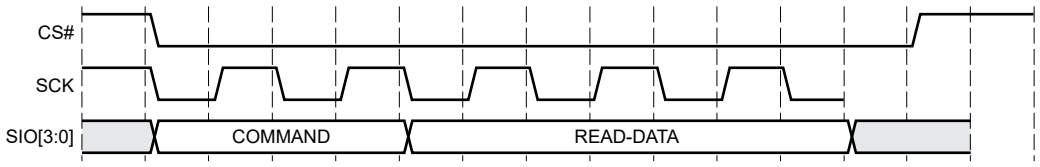


Figure 43. Unsupported read behavior

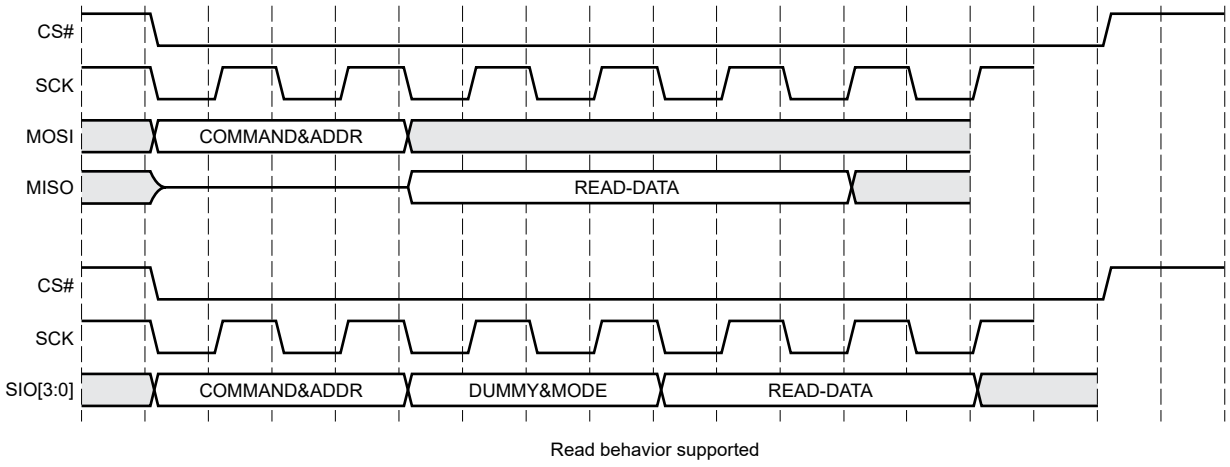


Figure 44. Supported read behavior

10.3.15.2 Receive clock source features

FlexSPI uses a read strobe to sample incoming data. There are several options for generating the read clock strobe that MCR0[RXCLKSRC] selects. Using the receive clock source method has implications for the board connections and can affect the maximum frequency that the interface supports. See the device data sheet for details on the maximum frequency supported for each RXCLKSRC option.

Table 86. RXCLKSRC options

MCR0[RXCLKSRC]	Description	Board connection	Max Frequency
0	Internal dummy read strobe and internal loopback	DQS pin is not used. The DQS pin can be configured for an alternate signal function.	Lowest
1	Internal dummy read strobe and loopback from DQS pad	FlexSPI uses the DQS pin, and it must be configured for its FlexSPI function. The internally generated read strobe is sent to the DQS pin and is sampled at the pin to match more closely the data pin timings. The DQS pin is typically left floating. External capacitance can be added to adjust timing.	Medium
2	SCK output and loopback from SCK pad	SCK is used as the read strobe. FlexSPI uses the signal looped back from the SCK pad as the read strobe to match more closely the data pin timings. DQS pin is not used and can be configured for an alternate signal function.	Medium
3	Flash-memory-provided read strobe	DQS pin is connected to the DQS or RWDS signal that the memory provides.	Highest

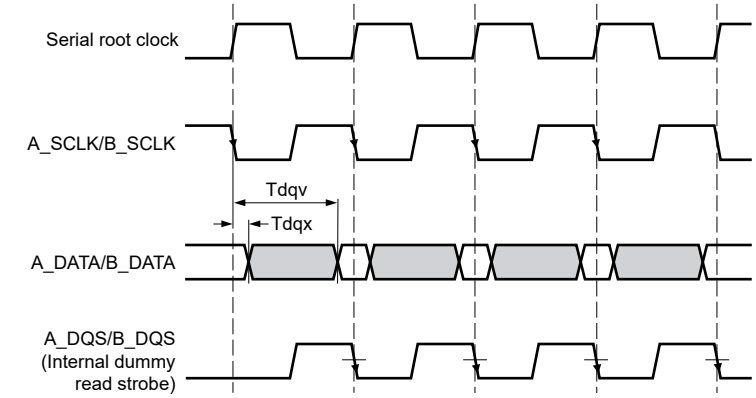
10.3.15.3 Input timing for sampling with dummy read strobe

This section describes the input timing when sampling with the internal dummy read strobe (MCR0[RXCLKSRC] = 0 or 1). The timing for sampling with an internal dummy read strobe loopback is very similar to the timing for loopback from pad. Sampling with a dummy read strobe loopback from the DQS pad can achieve a higher read frequency. It compensates for the delay of the SCLK output path and data pin input path.

The input timing is different for SDR mode and DDR mode.

- **Input timing for sampling with dummy read strobe in SDR mode**

  For SDR read or learn instructions, FlexSPI samples input data pins with the falling edge of the dummy read strobe. Figure 45 shows the input timing for sampling with dummy read strobe in SDR mode.



Tdqv is the setup time from SCLK to the next valid data.
Tdqx is the hold time from SCLK to the current valid data.

Figure 45. Input timing for sampling with dummy read strobe in SDR mode

- **Input timing for sampling with dummy read strobe in DDR mode**

  For DDR read or learn instructions, FlexSPI samples input data pins on rising and falling edges of the dummy read strobe. Figure 46 shows the input timing for sampling with dummy read strobe in DDR mode.

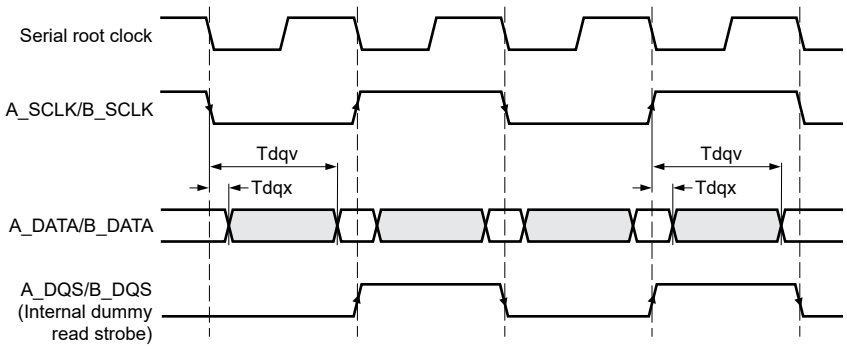


Figure 46. Input timing for sampling with dummy read strobe in DDR mode

10.3.15.4 Input timing for sampling with SCK output looped back from SCK pad

This section describes the input timing when sampling with the SCK output looped back from the SCK pad (MCR0[RXCLKSRC] = 2). The input timing is similar to sampling with a dummy read strobe. SCK output toggles for all types of instructions, but internal dummy strobe only toggles for read and learn instructions. In this case, FlexSPI receives more data bits when sampling with SCK output. FlexSPI automatically ignores the redundant data bits sampled.

10.3.15.5 Input timing for sampling with flash-memory-provided read strobe

This section describes the input timing when sampling with flash-memory-provided read strobe (MCR0[RXCLKSRC] = 3). The input timing is different for SDR mode and DDR mode.

NOTE

There are no known devices that provide a read strobe and support SDR mode operation.

- **Flash-memory-provided read strobe with SCLK**

  Certain flash devices provide both read data and read strobes with SCLK. Then the read strobe edge is aligned with the read data change. The FlexSPI controller delays the read strobe for one half cycle of the serial root clock (with DLL), then samples

read data with the delayed strobe. See DLL configuration for sampling. Figure 47 and Figure 48 show the input timing for sampling with flash memory read strobe in SDR mode and DDR mode.

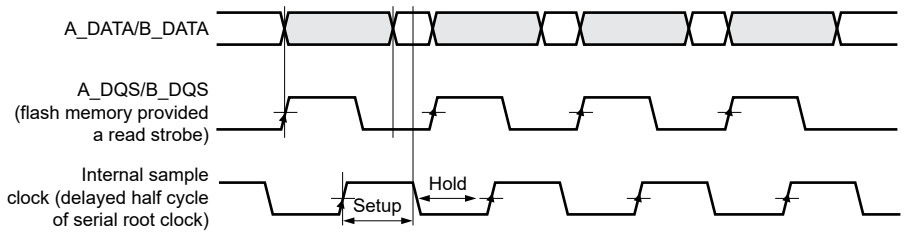


Figure 47. Input timing 2 for flash-memory-provided read strobe in SDR mode

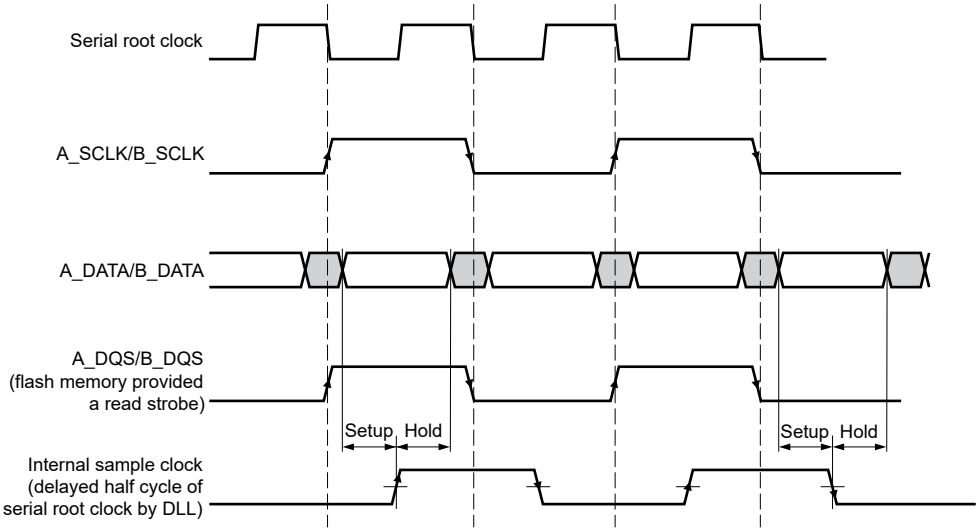


Figure 48. Input timing 2 for flash-memory-provided read strobe in DDR mode

10.3.15.6 DLL configuration for sampling

For the four sampling clock sources described previously, the input timing differs depending on the DLLxCR register configuration, according to the sampling clock source mode. The DLL is a delay-line chain that can be set to a fixed number of delay cells. It can also be auto-adjusted to lock on a certain phase delay to the reference clock.

- DLLxCR must be set to 0000_0100h (one fixed delay cell in DLL delay chain) in the following cases:
  - Sampling data with dummy read strobe looped back internally (MCR0[RXCLKSRC] = 0h)
  - Sampling data with dummy read strobe looped back from DQS pad (MCR0[RXCLKSRC] = 1h)
  - Sampling data with SCLK output looped back from SCLK pad (MCR0[RXCLKSRC] = 2h)
- When data is sampled with a flash-memory-provided read strobe (MCR0[RXCLKSRC] = 3h) and flash memory provides the read strobe with SCLK, DLL must be configured as follows (when serial root clock is over 100 Mhz. These settings ensure a lock of the reference clock (serial root clock).
  - DLLxCR[SLVDLYTARGET] is up to flash parameters used.
  - DLLxCR[REFPHASEGAP] = 2h
  - DLLxCR[DLLEN] = 1h
  - DLLxCR[OVRDEN] = 0h
  - Other fields in DLLxCR must be kept at their reset values (all zeroes).

After the register settings above are completed, set the DLL with following steps.

1. Reset the DLL(DLLxCR[DLLRESET] or MCR0[SWRESET]).
2. Wait for STS2[xREFLOCK] and STS2[xSLVLOCK] being asserted.
3. Wait for equal or more than 512 serial root clock cycles.
4. Start read or write operation.

NOTE

If the serial root clock is slower than 100 MHz, DLL cannot lock on a half cycle of serial root clock. The delay cell number is limited in the delay chain. DLL must be configured as follows instead:

- DLLxCR[OVRDEN] = 1
- DLLxCR[OVRDVAL] = N

  Each delay cell in DLL is about 75 ps to 225 ps. The delay of DLL delay chain is (N × Delay_cell_delay). N is set based on the maximum DDR frequency (N ≤ equation). This value is calculated based on the MAX_DDR_FREQ parameter. N = 34. This value is a recommended value. If a failure occurs, the value might require adjustment in a real application.

  You can calculate the delay of one delay cell and obtain the value of N. If the serial root clock is lower than 100 MHz, the delay chain can't lock with DLLxCR[SLVDLYTARGET] = 0xF (half cycle of root clock delay). You need to set DLLxCR[SLVDLYTARGET] as 0xE (15/32 cycle delay) or less. When DLL is locked (read STS2[AREFLOCK] and STS2[ASLVLOCK]), the delay cell number = STS2[ASLVSEL] +1 and the delay value (one delay cell) = (total delay)/(delay cell number). Based on delay value, the N can be easily calculated.
- Other fields in DLLxCR must be kept at their reset values (all zeroes).

10.3.16 Data learning

The FlexSPI controller generates 16 clock phases (Phase 0–15) via delay cell line (DCL) with the selected sample clock. Clock phase 0 is the selected sample clock (DQS_IN) with no delay cell. 16 sampling blocks are implemented for both Port A and 16 sampling blocks are implemented for Port B.

During a learn instruction, FlexSPI compares the sampled data bits with the internal data learn pattern in the Data Learning Pattern (DLPR) register and determines the correct sampling clock phase. The phase selection is automatically updated after a learn instruction and applied to subsequent read instructions or sequences.

When data learning is disabled (MCR0[LEARNEN] = 0), FlexSPI always uses clock phase 0 to sample FlexSPI data lines. Attempting to execute a learn instruction results in an error flag (INTR[IPCMDERR] or INTR[AHBCMDERR]).

FlexSPI supports data learning in both SDR mode and DDR mode. Data learning is not supported if the sampling clock source is a flash-memory-provided read strobe (MCR0[RXCLKSRC] = 3h).

Data learning can achieve high read frequency, but the flash memory input timing still limits the highest frequency. Flash memory must receive the command code and flash address bits.

When data learning is enabled, FlexSPI uses clock phase 0 after reset and before executing a learn instruction. The clock phase selection is updated after FlexSPI executes the learn instruction. The selected sample clock phase can be polled via STS0[DATALEARNPHASEA] and STS0[DATALEARNPHASEB]. When data learning fails, the INTR[DATALEARNFAIL] interrupt flag is set, and the previous clock phase selection is kept. You can write 1 to MCR2[CLRLEARNPHASE] to reset the internal clock phase selection to clock phase 0.

10.3.16.1 Data learning with flash-memory-provided preamble bit

Certain flash devices support driving with preamble bits before driving read data in each read command sequence. Preamble bits are also called the data-learning pattern (DLP). The DLP is programmable via configuration register in the flash device.

For the specified read command sequence, flash memory returns preamble bits after dummy cycles and before returning read data.

For these flash devices, the operation flow with data learning is:

1. Configure the data learning pattern in Data Learning Pattern (DLPR) register.
2. Configure the same data learning pattern in the flash device by triggering an IP command.
3. Enable data learning in the flash device by triggering an IP command.
4. Configure LUT sequence with valid read command sequence containing the learn instruction.
5. Trigger a flash memory read command via AHB or IP command as normal.

NOTE

The first four steps are executed only once. They are not needed for every flash memory read command.

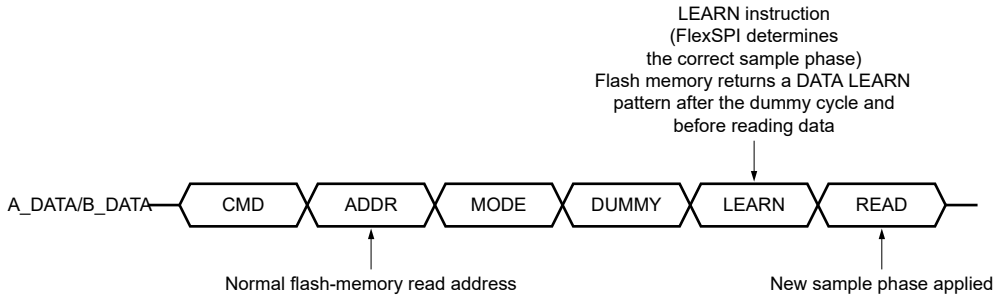


Figure 49. Data learning flow with flash-memory-provided preamble bit

10.3.16.2 Data learning without flash-memory-provided preamble bit

For flash devices that do not provide a preamble bit, the DLP cannot be returned automatically for each read command sequence. For these flash devices, the data learning sequence is:

1. Set data learning pattern in Data Learning Pattern (DLPR) register.
2. Reserve a certain flash memory area and program data (according to data learning pattern and flash memory access mode) to this area via IP command.
3. Configure LUT sequence with read sequence using a learn instruction instead of a read instruction. Trigger the read sequence to the reserved flash memory area via IP command.
4. Trigger the flash memory read command via AHB or IP command as normal. No learn instruction is needed in this read sequence.

NOTE

- The first three steps are executed only once. They are not needed for every flash memory read command.
- Step 4 should be executed with a certain interval to ensure that the internal sampling clock phase is adjusted in time.

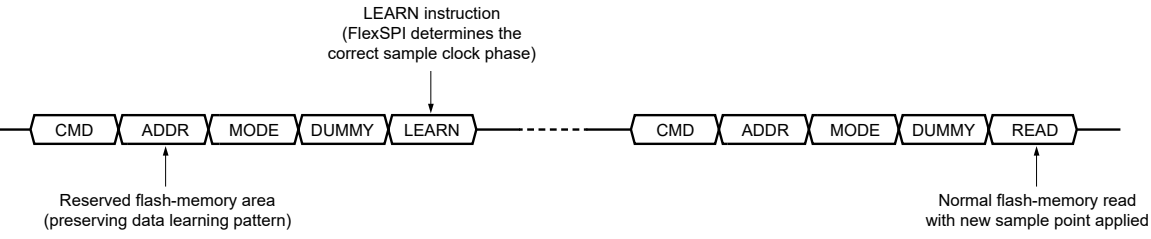


Figure 50. Data learning flow without flash-memory-provided preamble bit

NOTE

- For flash devices that do not provide a preamble bit, software overhead is required to read the reserved flash memory area at the interval.

These items determine the preprogramming data:

- Data Learning Pattern (DLPR register setting)
- DLP bit length
- Individual or Parallel mode for read command
- Octal, Quad, Dual, or Single mode for read command

Figure 51 shows a preprogramming data example where the DLPR value is 43h, eight bits, and flash memory is read in Quad mode and Parallel mode.

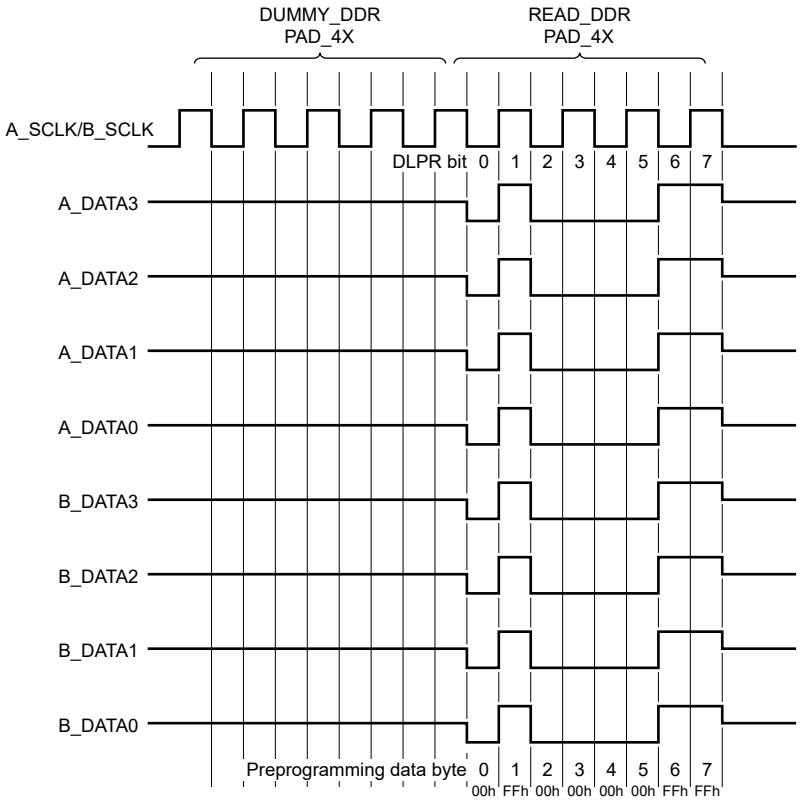


Figure 51. Preprogramming data for data learning

10.3.17 Execute-In-Place (XIP) enhanced mode

FlexSPI always supports XIP, regardless of whether external devices provide XIP enhanced mode. To support XIP, put program codes on an external device, then read or execute these codes directly via AHB read access to serial flash memory space. There is no configuration or status polling needed during AHB read access to the memory of the external device. The AHB receive buffer is fully transparent to software.

Certain devices provide XIP enhanced mode to improve code execution. In this mode, the command code is only sent in the first read instruction. It saves many cycles for command instructions and improves code execution. Devices enter or exit XIP enhanced mode via a special sequence that external devices specify. See the external device data sheet for more details.

Normally, a device enters XIP enhanced mode via this sequence:

1. Enable XIP enhanced mode in external flash device via IP command.

2. Send the first read sequence to external flash device with correct mode bits. Command code is included in this read sequence.
3. Send the subsequent read sequences to external flash device with correct mode bits. Command code is not included in these read sequences. Mode bits must be sent according to the flash specified; otherwise, the flash exits XIP enhanced mode.

The instruction JMP_ON_CS in FlexSPI must be used to support XIP enhanced mode in external devices. This instruction is only allowed in the AHB read command; otherwise, the instruction generates an IPCMDERR or AHBCDMERR interrupt (when the interrupt is enabled). When external devices do not support XIP enhanced mode, JMP_ON_CS must never be used. To support XIP enhanced mode:

- The first instruction in the read sequence must be the command instruction.
- The last valid instruction must be JMP_ON_CS (with operand 1h).

For the first AHB read command triggered, FlexSPI executes the instructions from the instruction pointer 0 in the sequence (which is the command instruction). After this sequence is executed, FlexSPI saves the operand in the JMP_ON_CS instruction as the start pointer for the next command to current device internally. For the next AHB read command triggered, FlexSPI executes from the instruction pointer 1h, bypassing the command instruction.

After XIP enhanced mode is started, it is possible to access flash via other commands like program commands. You must, however, use a special sequence to exit enhanced XIP mode before you can run a different command. Figure 52 indicates XIP operation with Flash XIP enhanced mode.

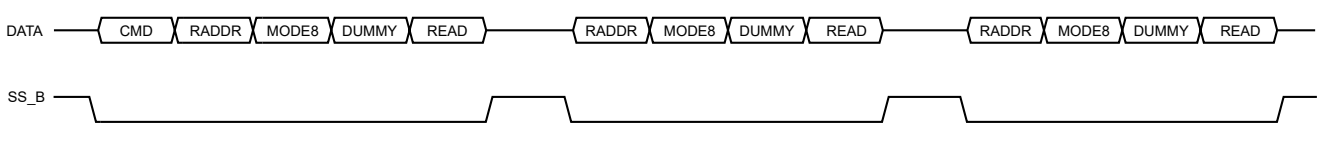


Figure 52. XIP Enhanced Mode Operation

10.3.18 AHB bus blocking

FlexSPI can temporarily block some or all AHB transactions. This feature is useful when IP commands are in progress that could cause a potential conflict with an AHB request. Examples of situations where IP bus blocking would be used are:

- Avoiding RWW (read while write). While programming or erasing flash, reads are not allowed. AHB bus blocking can prevent the AHB bus from requesting a read while IP write commands are in progress.
- Decrypting while encrypting. The IPED of FlexSPI cannot support simultaneous encryption and decryption. Partial AHB bus blocking can prevent AHB reads from IPED regions while IP writes to encrypted block are in progress.

To block all AHB transactions:

1. Configure blocking for all AHB transactions by writing 1 to IPCR2[IPBLKALLAHB].
2. Request AHB bus blocking by writing 1 to IPCR2[IPBLKAHBREQ].
3. Wait for the AHB bus blocking request to be acknowledged (IPCR2[IPBLKAHBACK] = 1).
4. Perform IP write commands as needed.
5. Request normal operation by writing 0 to IPCR2[IPBLKAHBREQ].
6. Wait for AHB blocking request clear to be acknowledged (IPCR2[IPBLKAHBACK] = 0).

To block AHB transactions to IPED regions only, while allowing AHB transactions to unencrypted areas:

1. Configure blocking for AHB transactions to IPED areas by writing 0 to IPCR2[IPBLKALLAHB].
2. Request AHB bus blocking by writing 1 to IPCR2[IPBLKAHBREQ].
3. Wait for the AHB bus blocking request to be acknowledged (IPCR2[IPBLKAHBACK] = 1).

4. Perform IP write commands as needed.
5. Request normal operation by writing 0 to IPCR2[IPBLKAHBREQ].
6. Wait for AHB blocking request clear to be acknowledged (IPCR2[IPBLKAHBACK] = 0).

NOTE

AHB bus blocking can only be used to prevent conflicts between AHB and IP transactions. Software must handle conflicts between multiple AHB transactions.

10.3.19 Domain control

FLEXSPI supports two kinds of protection modes to manage nonsecure controller IP command access:

- IP command trigger
- LUT write

The chip must maintain domain control of the AHB region.

10.3.19.1 IP command domain control

Two pairs of start and end registers can be configured to specify the regions that allow nonsecure controller IP command access: IPSNSZSTARTn and IPSNSZENDn. Only secure controllers can write to these registers. Nonsecure controllers cannot configure or change the regions where IP commands can be used. Any addresses outside the nonsecure regions are treated as secure.

NOTE

The end address must be larger than the start address. The start address must be the flash memory address.

Once the address range of these registers becomes valid (start address < end address), any IP command issued by nonsecure controller targeting a secure region is blocked. No IP command is triggered. The INTR[IPCMDSECUREVIO] interrupt is set instead.

The program sequence is:

1. Write start address to IPSNSZSTART0 and end address to IPSNSZEND0.
2. Write start address to IPSNSZSTART1 and end address to IPSNSZEND1.

After the IPSNSZSTARTn and IPSNSZENDn registers are configured, FlexSPI responds to IP commands (IPCMD[TRG] = 1) as shown in Table 87.

Table 87. IPS access

Secure type: ips_nonsecure = 0 and ips_supervisor_access = 1	IPCR0[SFAR] address located within NSZ range	IPCMD successfully triggered
	IPCR0[SFAR] address outside NSZ range	IPCMD successfully triggered
Nonsecure type: ips_nonsecure = 1 or ips_supervisor_access = 0	IPCR0[SFAR] address located within NSZ range	IPCMD successfully triggered
	IPCR0[SFAR] address located outside NSZ range	IPCMD failed to trigger, INTR[IPCMDSECUREVIO] set

10.3.19.2 LUT write protection

LUTCR[PROTECT] controls LUT write protection. By default, no protection is applied on the LUT Control (LUTCR) register and LUT memory, and any controller can access the LUT freely.

When LUTCR[PROTECT] = 1, only a secure controller can change the value of LUTCR and the content of the LUT table. Unless hardware resets FlexSPI to return to the LUT write-protection-disabled state, nonsecure controllers cannot write to the LUT table or LUTCR register.

No error or interrupt occurs if a non-secure controller attempts to write to the protected LUT; the write operation is ignored.

10.4 External signals

This section provides the external signal information for the FlexSPI module.

Table 88. External Signal List

Signal name	Function	Direction	Description
A_SS0_B	Peripheral Chip Select Flash A1	O	Chip select for the serial flash device A1
A_SS1_B	Peripheral Chip Select Flash A2	O	Chip select for the serial flash device A2
B_SS0_B	Peripheral Chip Select Flash B1	O	Chip select for the serial flash device B1
B_SS1_B	Peripheral Chip Select Flash B2	O	Chip select for the serial flash device B2
A_SCLK	Serial Clock Flash A	O	Serial clock output to the serial flash device A. This clock runs at half the frequency of serial clock root in DDR mode, and at the same frequency as serial clock root in SDR mode. Clock output toggles during the entire flash memory access sequence.
B_SCLK	Serial Clock Flash B	O	Serial clock output to the serial flash device B. This clock runs at half the frequency of serial clock root in DDR mode, and at the same frequency as serial clock root in SDR mode. Clock output toggles during the entire flash memory access sequence. **NOTE** B_SCLK may be used as the differential clock output of A_SCLK by writing 1 to MCR2[SCKBDIFFOPT].
A_DATAn	Serial I/O Flash A	I/O	Data I/O lines to and from the serial flash device A
B_DATAn	Serial I/O Flash B	I/O	Data I/O lines to and from the serial flash device B
A_DQS	Data Strobe Signal Flash A	I/O	Data strobe signal for port A. This signal has three functions: • Driven with read strobe by external device. Some flash devices provide the read strobe signal together with read data. In this case, if the external device drives this pad only

Table continues on the next page...

Table 88. External Signal List (continued)

Signal name	Function	Direction	Description
			when reading flash memory data, this pad may require a **pull-down** resistor. • Driven with latency information by external device. Some devices use this pin to indicate the dummy cycles needed (before program or read data transfer) such as HyperRAM or HyperFlash. • Loopback dummy read strobe. The FlexSPI controller provides internal dummy read strobe for flash memory read data. Higher read frequency can be achieved by looping back this dummy read strobe from the pad. This pin can be floated or have some capacitive load added at the board level to compensate for load on DATA or SCLK pins.
B_DQS	Data Strobe Signal Flash B	I/O	Similar to A_DQS

10.5 Initialization

The FlexSPI controller initialization sequence is:

1. Enable controller clocks (AHB clock, IP bus clock, or serial root clock) at system level.
2. To enter module stop mode, write 1 to MCR0[MDIS].
3. Configure module control registers: MCR0, MCR1, and MCR2. Do not change MCR0[MDIS].
4. When AHB commands are used, configure AHB Bus Control (AHBCR) and AHB Receive Buffer Control (AHBRXBUFxCR0) registers.
5. Configure flash control registers (FLSHxCR0, FLSHxCR1, or FLSHxCR2) according to external device type.
6. Configure DLL control register (DLLxCR) according to sample clock source selection.
7. To exit module stop mode, write 0 to MCR0[MDIS].
8. Configure LUT as needed for AHB command or IP command.
9. Optionally, to reset controller, write 1 to MCR0[SWRESET].

External devices must be configured via IP command normally after initialization. For example, the WRITE STATUS command performs device configuration for most serial NOR flash memory.

10.6 Application information

This section describes applications that FlexSPI supports.

10.6.1 Application on serial NOR flash device

This section provides the example LUT instruction sequences for serial NOR flash device (Cypress Flash S25FS128S).

10.6.1.1 Write enable command

Table 89 shows the WRITE ENABLE command sequence.

Table 89. WRITE ENABLE command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h	06h	Command name: WREN
1-7	STOP (00h)	0h	00h	

10.6.1.2 Write registers command

Table 90 shows the Write Registers command sequence.

Table 90. Write Registers command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	01h	Command name: WRR
1	WRITE_SDR	0h (Single)	1h or 2h	Data size is 1 byte or 2 bytes. Byte 0 is write data for Status Register 1; Byte 1 is write data for Configuration Register 1. IPCR1[IDATSZ] can override this value.
2-7	STOP (00h)	0h	00h	

10.6.1.3 Page Program command

Table 91 shows Page Program command sequence.

Table 91. PAGE PROGRAM command (Cypress serial NOR flash)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	02h or 12h	Command name: PP or 4PP PP is 3-byte address mode. 4PP is 4-byte address mode.
1	ADDR_SDR	0h (Single)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	WRITE_SDR	0h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default programming data size. This value is ignored for AHB command.
3-7	STOP (00h)	0h	00h	

Table 92 shows Page Program command sequence (QPI mode).

Table 92. PAGE PROGRAM (QPI mode) command (Cypress serial NOR flash)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	2h (Quad)	02h or 12h	Command name: PP or 4PP PP is 3-byte address mode. 4PP is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	WRITE_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default programming data size. This value is ignored for AHB command.
3-7	STOP (00h)	0h	00h	

10.6.1.4 Read Status 1 command

Table 93 shows READ STATUS 1 command sequence.

Table 93. READ STATUS 1 command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	05h	Command name: RDSR1
1	READ_SDR	0h (Single)	1h	1 byte for Status register 1
2-7	STOP (00h)	0h	00h	

10.6.1.5 Read command

Table 94 shows READ command sequence.

Table 94. READ command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	03h or 13h	Command name: READ or 4READ READ is 3-byte address mode. 4READ is 4-byte address mode.
1	ADDR_SDR	0h (Single)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	READ_SDR	0h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
3-7	STOP (00h)	0h	00h	

10.6.1.6 Fast Read command

Table 95 shows Fast Read command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[3:0] = 8h.

Table 95. FAST_READ command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	0Bh or 0Ch	Command name: FAST_READ or 4FAST_READ FAST_READ is 3-byte address mode. 4FAST_READ is 4-byte address mode.
1	ADDR_SDR	0h (Single)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	DUMMY_SDR	0h (Single)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
3	READ_SDR	0h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
4-7	STOP (00h)	0h	00h	

10.6.1.7 Dual IO Fast Read command

Table 96 shows Dual IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[3:0] = 8h, Continuous Read mode.

Table 96. Dual IO FAST_READ command (Continuous Read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	BBh or BCh	Command name: DIOR or 4DIOR DIOR is 3-byte address mode. 4DIOR is 4-byte address mode.
1	ADDR_SDR	1h (Dual)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	1h (Dual)	Axh	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_SDR	1h (Dual)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	1h (Dual)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.

Table continues on the next page...

Table 96. Dual IO FAST_READ command (Continuous Read mode) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
5	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
6-7	STOP (00h)	0h	00h	

Table 97 shows Dual IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[3:0] = 8h, Noncontinuous Read mode.

Table 97. Dual IO FAST_READ command (Noncontinuous Read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	BBh or BCh	Command name: DIOR or 4DIOR DIOR is 3-byte address mode. 4DIOR is 4-byte address mode.
1	ADDR_SDR	1h (Dual)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	1h (Dual)	Any value other than Axh	Exit continuous read mode or remain in noncontinuous read mode.
3	DUMMY_SDR	1h (Dual)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	1h (Dual)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5-7	STOP (00h)	0h	00h	

10.6.1.8 Quad IO Fast Read command

Table 98 shows Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Noncontinuous read mode.

Table 98. Quad IO FAST_READ command (Non-QPI mode, Noncontinuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EBh or ECh	Command name: QIOR or 4QIOR QIOR is 3-byte address mode. 4QIOR is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.

Table continues on the next page...

Table 98. Quad IO FAST_READ command (Non-QPI mode, Noncontinuous read mode) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
2	MODE8_SDR	2h (Quad)	Any value other than Axh	Exit continuous read mode or remain in noncontinuous read mode.
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5-7	STOP (00h)	0h	00h	

Table 99 shows Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Continuous read mode.

Table 99. Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EBh or ECh	Command name: QIOR or 4QIOR QIOR is 3-byte address mode. 4QIOR is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
6-7	STOP (00h)	0h	00h	

Table 100 shows Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 1, CR2V[3:0] = 8h, QPI mode, Continuous read mode.

Table 100. Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	2h (Quad)	EBh or ECh	Command name: QIOR or 4QIOR

Table continues on the next page...

Table 100. Quad IO FAST_READ command (QPI mode, Continuous read mode) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				QIOR is 3-byte address mode. 4QIOR is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
6-7	STOP (00h)	0h	00h	

10.6.1.9 DDR Quad IO Fast Read command

Table 101 shows DDR Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Non-continuous read mode.

Table 101. DDR Quad IO FAST_READ command (Non-QPI mode, Noncontinuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EDh or EEh	Command name: QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-byte address mode. 4QIOR_DDR is 4-byte address mode.
1	ADDR_DDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_DDR	2h (Quad)	Any value other than Axh	Exit continuous read mode or keep in noncontinuous read mode.
3	DUMMY_DDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	LEARN_DDR	2h (Quad)	1h	DLP is 8 bits.
5	READ_DDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading

Table continues on the next page...

Table 101. DDR Quad IO FAST_READ command (Non-QPI mode, Noncontinuous read mode) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				data size. This value is ignored for AHB command.
6-7	STOP (00h)	0h	00h	

Table 102 shows DDR Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Continuous read mode.

Table 102. DDR Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EDh or EEh	Command name: QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-byte address mode. 4QIOR_DDR is 4-byte address mode.
1	ADDR_DDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_DDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_DDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	LEARN_DDR	2h (Quad)	1h	DLP is 8 bits.
5	READ_DDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
6	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
7	STOP (00h)	0h	00h	

Table 103 shows DDR Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 1, CR2V[3:0] = 8h, QPI mode, Continuous read mode.

Table 103. DDR Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	2h (Quad)	EDh or EEh	Command name: QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-byte address mode. 4QIOR_DDR is 4-byte address mode.

Table continues on the next page...

Table 103. DDR Quad IO FAST_READ command (QPI mode, Continuous read mode) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
1	ADDR_DDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_DDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_DDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	LEARN_DDR	2h (Quad)	1h	DLP is 8 bits.
5	READ_DDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
6	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
7	STOP (00h)	0h	00h	

10.6.2 Application on HyperBus device

This section provides the example LUT instruction sequences for HyperBus device (Cypress RPC flash, HyperRam, or HyperFlash).

10.6.2.1 HyperFlash

This section provides example sequences for HyperFlash devices (Cypress S26KS series).

Table 104 shows the read status command sequence.

Table 104. Read status command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr = 555h, Data = 70h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W# = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	05h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 0070h

Table continues on the next page...

Table 104. Read status command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	7	CMD_DDR	3h (Octal)	70h	
1 (Read - Addr = xxx, Data = Status register data)	0	CMD_DDR	3h (Octal)	A0h	CA bit 47: (R/W#) = 1h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
	2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS_DDR	3h (Octal)	0Bh	When latency count = 11
	4	READ_DDR	0h (Octal)	4h	4-Byte read
	5-7	STOP (00h)	0h	00h	

Table 105 shows the read (memory) command sequence.

Table 105. Read (memory) command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Read - Addr = xxx, Data = memory data)	0	CMD_DDR	3h (Octal)	A0h	CA bit 47: (R/W#) = 1h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
	2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS_DDR	3h (Octal)	0Bh	When latency count = 11
	4	READ_DDR	3h (Octal)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
	5-7	STOP (00h)	0h	00h	

Table 106 shows the word program command sequence.

Table 106. Word program command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr = 555h, Data = AAh)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	05h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 00AAh
	7	CMD_DDR	3h (Octal)	AAh	
1 (Write - Addr = 2AAh, Data = 55h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 000055h (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	55h	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 02h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	02h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 0055h
	7	CMD_DDR	3h (Octal)	55h	
2 (Write - Addr = 555h, Data = A0h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	55h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 00A0h

Table continues on the next page...

Table 106. Word program command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	7	CMD_DDR	3h (Octal)	A0h	
3 (Word Program)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
	2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
	3	WRITE_DDR	3h (Octal)	02h	2-Byte written data
	4-7	STOP (0h)	0h	00h	

Table 107 shows Write-to-Buffer and Program-Buffer-to-Flash command sequence.

Table 107. Write-to-Buffer and Program-Buffer-to-Flash command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr = 555h, Data = AAh)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	05h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 00AAh
	7	CMD_DDR	3h (Octal)	AAh	
1 (Write - Addr = 2AAh, Data = 55h)	0	CMD_DDR	0h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 000055h (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	55h	

Table continues on the next page...

Table 107. Write-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	4	CMD_DDR	3h (Octal)	00h	Column Address: 02h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	02h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 0055h
	7	CMD_DDR	3h (Octal)	55h	
2 (Write - Addr = SA, Data = 25h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: SA (24 bit) SA is sector address. Write SA to IPCR0[SFAR].
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	3h (Octal)	00h	Write Data: 0025h
	4	CMD_DDR	3h (Octal)	25h	
2 (Write - Addr = SA, Data = WC)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: SA (24 bit) SA is sector address. Write SA to IPCR0[SFAR].
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	3h (Octal)	WC	Write Data: WC WC is word count.
	4	CMD_DDR	3h (Octal)		
3 - N (Write - Addr = WBL, Data = PD) N is the word count + 2	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: WBL (24 bit) WBL is write buffer location. Write WBL to IPCR0[SFAR].

Table continues on the next page...

Table 107. Write-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	WRITE_DDR	3h (Octal)	02h	2-Byte write data
	4-7	STOP (0h)	0h	00h	
N + 1 (Write - Addr = SA, Data = 29) Program Buffer to Flash	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: SA (24 bit) SA is sector address. Write SA to IPCR0[SFAR].
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	3h (Octal)	00h	Write Data: 29h
	4	CMD_DDR	3h (Octal)	29h	
	5-7	STOP (0h)	0h	00h	

10.6.2.2 HyperRAM

This section provides example sequences for HyperRAM (Cypress S27KL series).

The Read (memory) command sequence is same as HyperFlash. Table 108 shows the Write (memory) command sequence.

Table 108. Write (memory) command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
3	DUMMY_RWDS_DDR	3h (Octal)	0Bh	When latency count = 11
4	WRITE_DDR	3h (Octal)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default writing

Table continues on the next page...

Table 108. Write (memory) command (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				data size. This value is ignored for AHB command.
5-7	STOP (00h)	0h	00h	

10.6.3 Application on Serial NAND flash device

This section provides example LUT instruction sequences for serial NAND flash device (Micron Flash MT29 series). The operation of serial NAND flash is similar to serial NOR flash.

The read operation sequence is:

- Page read. Transfer the data from the NAND flash array to the cache register.
- Get feature to read the status.
- Random data read

Table 109 shows the page read command sequence.

Table 109. Page Read command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	13h	Command code: 13h
1	RADDR_SDR	1h (Single)	18h	Row address: 24 bit
2-7	STOP (0h)	0h	00h	

Table 110 shows Get Feature command sequence.

Table 110. Get Feature command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	0Fh	Command code: 0Fh
1	CMD_SDR	1h (Single)	C0h	Status register address (C0h)
2	READ_SDR	0x1 (Single)	02h	2 bytes read data
3-7	STOP (0h)	0h	00h	

Table 111 shows Random Data Read command sequence.

Table 111. Read from Cache x4 command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	6Bh	Command code: 6Bh
1	MODE4_SDR	1h (Single)	0h or 1h	If plane selection is one, software should decode the flash address and write 1h to mode bits. If plane selection

Table continues on the next page...

Table 111. Read from Cache x4 command (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				is zero, software should write 0h to mode bits. Plane selection bit is 18th bit of flash address. If NAND flash size is less than 4 Gbit, plane selection is always zero.
2	CADDR_SDR	1h (Single)	0Ch	Column address: 12 bit
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle number: 8 (serial root clock)
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size.
5-7	STOP (0x0)	0x0	0x00	

The program operation sequence is:

- Write enable
- Program Load. Transfer the write data to the cache register.
- Program Execute
- Get Feature to read the status.

Table 112 shows Program Load command sequence.

Table 112. Program Load command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	02h	Command code: 02h
1	MODE4_SDR	1h (Single)	0h or 1h	If plane selection is one, software should decode the flash address and write 1h to mode bits. If plane selection is zero, software should write 0h to mode bits. Plane selection bit is 18th bit of flash address. If NAND flash size is less than 4 Gbit, plane selection is always zero.
2	CADDR_SDR	1h (Single)	0Ch	Column address: 12 bit
3	WRITE_SDR	1h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default writing data size.
4-7	STOP (0h)	0h	00h	

Table 113 shows Program Execute command sequence.

Table 113. Program Execute command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	10h	Command code: 10h
1	RADDR_SDR	1h (Single)	18h	Row Address: 24 bit
2-7	STOP (0h)	0h	00h	

10.6.4 Application on FPGA device

An FPGA device can be accessed via both AHB and IP commands. All AHB accesses to FPGA are transparent to the software driver (no software intervention). An FPGA device may have some special requirements.

Table 114. Special requirements for FPGA devices

Condition	How to manage the condition
Device type may be different on A1, A2, B1, or B2.	Write 0 to MCR2[SAMEDEVICEEN] and configure the FLSHxCR0 and FLSHxCR1 registers separately for up to four external devices.
Device requires different wait cycle for programming.	The AHB write wait cycle number can be set separately for these four external devices (via FLSHxCR2[AWRWAIT]). Software can configure the sequences in LUT with different dummy instructions (operand determines dummy cycle). FlexSPI holds the AHB bus ready for this wait time; AHB bus performance may decrease when this wait time is very long.
Device requires different wait cycle for reading.	The AHB read sequence index and sequence number can be set separately for these four external devices (via FLSHxCR2[ARDSEQID] and FLSHxCR2[ARDSEQNUM]). Software can configure the sequences in LUT with different dummy instructions (operand determines dummy cycle).
Device may be sensitive to read instruction clock cycle number.	If its internal memory is implemented similar to a FIFO, the device is sensitive to the read instruction clock cycle number. Software can send the data size information to the external device via DATSZ instruction. The FPGA device decodes the data size information and determines how many data bytes must be popped.
Device may require interval time between chip select valid.	This interval can be managed via FLSHxCR1[CSINTERVAL].
Device may use SCLK as reference clock for its internal PLL.	SCLK must be free-running and the clock frequency must be stable. To achieve this configuration, write 1 to MCR0[SCKFREERUNEN] and only use SDR sequences.

10.6.5 Overview of error categories, flags, and triggered sources

Table 115 provides an overview of the categories, flags, and triggered sources of errors in FlexSPI.

Table 115. Error category, triggered sources, and flags

Error Category	Triggered Sources	Description	Error Flags
Command grant error	AHB write command	Command grant timeout	INTR[AHBCMDGE] is set AHB bus error response

Table continues on the next page...

Table 115. Error category, triggered sources, and flags (continued)

Error Category	Triggered Sources	Description	Error Flags
	AHB read command		INTR[AHBCMDGE] is set AHB bus error response
	IP command		INTR[IPCMDGE] is set
Command check error	AHB write command	• AHB write command with JMP_ON_CS instruction used in the sequence • Unknown instruction opcode in the sequence • Instruction DUMMY_SDR or DUMMY_RWDS_SDR used in DDR sequence • Instruction DUMMY_DDR or DUMMY_RWDS_DDR used in SDR sequence	INTR[AHBCMDERR] is set Command is not executed when an error is detected in command check
	AHB read command	• Unknown instruction opcode in the sequence • Instruction DUMMY_SDR or DUMMY_RWDS_SDR used in DDR sequence • Instruction DUMMY_DDR or DUMMY_RWDS_DDR used in SDR sequence	INTR[AHBCMDERR] is set Command is not executed when an error is detected in command check
	IP command	• IP command with JMP_ON_CS instruction used in the sequence • Unknown instruction opcode in the sequence • Instruction DUMMY_SDR or DUMMY_RWDS_SDR used in DDR sequence • Instruction DUMMY_DDR or DUMMY_RWDS_DDR used in SDR sequence • Flash boundary across	INTR[IPCMDERR] is set Command is not executed when an error is detected in command check
Command execution error	AHB write command	Command timeout during execution	INTR[AHBCMDERR] is set INTR[SEQTIMEOUT] is set An AHB bus error response

Table continues on the next page...

Table 115. Error category, triggered sources, and flags (continued)

Error Category	Triggered Sources	Description	Error Flags
			occurs, except in following cases: • Flush triggers AHB write command (INCR burst ended with AHB_TX_BUF not empty) • AHB bufferable write access and bufferable enabled (AHBCR[BUFFERABLEEN] = 1)
	AHB read command		INTR[AHBCMDERR] is set INTR[SEQTIMEOUT] is set AHB bus error response
	IP command		INTR[IPCMDERR] is set INTR[SEQTIMEOUT] is set
AHB bus timeout	AHB write command	AHB bus timeout (no bus ready return)	INTR[AHBBUSTIMEOUT] is set AHB bus error response
	AHB read command		
Data learning failed	Any command	No valid sample clock phase found after learn instruction is executed	INTR[DATALEARNFAIL] is set
Security error	IP command	Nonsecure controller requests IP command access to a secure region	INTR[IPCMDSECUREVIO] is set

10.7 Memory map and register definition

This section includes the FlexSPI module memory map and detailed descriptions of all registers.

10.7.1 Register access

All registers can be accessed with 8-bit, 16-bit, and 32-bit width operations. Never change the values of reserved fields in control registers. Changing the values of reserved fields may impact the normal functioning of the controller.

NOTE

When FlexSPI can access sensitive memory contents, the FlexSPI controller should protect those contents using resource isolation such as XRDC or TrustZone, or equivalent partition control via SCFW. Ensure that only trusted software is allowed to access the FlexSPI controller register interface.

10.7.2 FlexSPI register descriptions

10.7.2.1 FlexSPI memory map

FLEXSPI0 base address: 400C_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Control 0 (MCR0)	32	RW	FFFF_8002h
4h	Module Control 1 (MCR1)	32	RW	FFFF_FFFFh
8h	Module Control 2 (MCR2)	32	RW	2000_81F7h
Ch	AHB Bus Control (AHBCR)	32	RW	0000_0018h
10h	Interrupt Enable (INTEN)	32	RW	0000_0000h
14h	Interrupt (INTR)	32	RW	0000_0000h
18h	LUT Key (LUTKEY)	32	RW	5AF0_5AF0h
1Ch	LUT Control (LUTCR)	32	RW	0000_0002h
20h	AHB Receive Buffer 0 Control 0 (AHBRXBUF0CR0)	32	RW	8000_0010h
24h	AHB Receive Buffer 1 Control 0 (AHBRXBUF1CR0)	32	RW	8001_0010h
28h	AHB Receive Buffer 2 Control 0 (AHBRXBUF2CR0)	32	RW	8002_0010h
2Ch	AHB Receive Buffer 3 Control 0 (AHBRXBUF3CR0)	32	RW	8003_0010h
30h	AHB Receive Buffer 4 Control 0 (AHBRXBUF4CR0)	32	RW	8004_0010h
34h	AHB Receive Buffer 5 Control 0 (AHBRXBUF5CR0)	32	RW	8005_0010h
38h	AHB Receive Buffer 6 Control 0 (AHBRXBUF6CR0)	32	RW	8006_0010h
3Ch	AHB Receive Buffer 7 Control 0 (AHBRXBUF7CR0)	32	RW	8007_0010h
60h	Flash Control 0 (FLSHA1CR0)	32	RW	0001_0000h
64h	Flash Control 0 (FLSHA2CR0)	32	RW	0001_0000h
68h	Flash Control 0 (FLSHB1CR0)	32	RW	0001_0000h
6Ch	Flash Control 0 (FLSHB2CR0)	32	RW	0001_0000h
70h	Flash Control 1 (FLSHA1CR1)	32	RW	0000_0063h
74h	Flash Control 1 (FLSHA2CR1)	32	RW	0000_0063h
78h	Flash Control 1 (FLSHB1CR1)	32	RW	0000_0063h
7Ch	Flash Control 1 (FLSHB2CR1)	32	RW	0000_0063h
80h	Flash Control 2 (FLSHA1CR2)	32	RW	0000_0000h
84h	Flash Control 2 (FLSHA2CR2)	32	RW	0000_0000h
88h	Flash Control 2 (FLSHB1CR2)	32	RW	0000_0000h
8Ch	Flash Control 2 (FLSHB2CR2)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
94h	Flash Control 4 (FLSHCR4)	32	RW	0000_00C3h
A0h	IP Control 0 (IPCR0)	32	RW	0000_0000h
A4h	IP Control 1 (IPCR1)	32	RW	0000_0000h
A8h	IP Control 2 (IPCR2)	32	RW	0000_0000h
B0h	IP Command (IPCMD)	32	RW	0000_0000h
B4h	Data Learning Pattern (DLPR)	32	RW	0000_0000h
B8h	IP Receive FIFO Control (IPRXFCR)	32	RW	0000_0000h
BCh	IP Transmit FIFO Control (IPTXFCR)	32	RW	0000_0000h
C0h	DLL Control 0 (DLLACR)	32	RW	0000_0100h
C4h	DLL Control 0 (DLLBCR)	32	RW	0000_0100h
E0h	Status 0 (STS0)	32	R	0000_0002h
E4h	Status 1 (STS1)	32	R	0000_0000h
E8h	Status 2 (STS2)	32	R	0100_0100h
ECh	AHB Suspend Status (AHBSPNDSTS)	32	R	0000_0000h
F0h	IP Receive FIFO Status (IPRXFSTS)	32	R	0000_0000h
F4h	IP Transmit FIFO Status (IPTXFSTS)	32	R	0000_0000h
100h - 17Ch	IP Receive FIFO Data a (RFDR0 - RFDR31)	32	R	See section
180h - 1FCh	IP TX FIFO Data a (TFDR0 - TFDR31)	32	W	0000_0000h
200h - 2FCh	Lookup Table a (LUT0 - LUT63)	32	RW	See section
420h	HADDR REMAP Start Address (HADDRSTART)	32	RW	0000_0000h
424h	HADDR REMAP END ADDR (HADDREND)	32	RW	0000_0000h
428h	HADDR Remap Offset (HADDROFFSET)	32	RW	0000_0000h
42Ch	IPED Function Control (IPEDCTRL)	32	RW	0000_0000h
430h	IPS Nonsecure Region 0 Start Address (IPSNSZSTART0)	32	RW	0000_0000h
434h	IPS Nonsecure Region 0 End Address (IPSNSZEND0)	32	RW	0000_0000h
438h	IPS Nonsecure Region 1 Start Address (IPSNSZSTART1)	32	RW	0000_0000h
43Ch	IPS Nonsecure Region 1 End Address (IPSNSZEND1)	32	RW	0000_0000h
440h	Receive Buffer Start Address of Region 0 (AHBBUFREGIONSTART0)	32	RW	0000_0000h
444h	Receive Buffer Region 0 End Address (AHBBUFREGIONEND0)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
448h	Receive Buffer Start Address of Region 1 (AHBBUFREGIONSTART1)	32	RW	0000_0000h
44Ch	Receive Buffer Region 1 End Address (AHBBUFREGIONEND1)	32	RW	0000_0000h
450h	Receive Buffer Start Address of Region 2 (AHBBUFREGIONSTART2)	32	RW	0000_0000h
454h	Receive Buffer Region 2 End Address (AHBBUFREGIONEND2)	32	RW	0000_0000h
458h	Receive Buffer Start Address of Region 3 (AHBBUFREGIONSTART3)	32	RW	0000_0000h
45Ch	Receive Buffer Region 3 End Address (AHBBUFREGIONEND3)	32	RW	0000_0000h
500h	IPED context control 0 (IPEDCTXCTRL0)	32	RW	5555_5555h
504h	IPED context control 1 (IPEDCTXCTRL1)	32	RW	AAAA_AAAAh
520h	IPED Context0 IV0 (IPEDCTX0IV0)	32	RW	0000_0000h
524h	IPED Context0 IV1 (IPEDCTX0IV1)	32	RW	0000_0000h
528h	Start Address of Region (IPEDCTX0START)	32	RW	0000_0000h
52Ch	End Address of Region (IPEDCTX0END)	32	RW	0000_0000h
540h	IPED Context1 IV0 (IPEDCTX1IV0)	32	RW	0000_0000h
544h	IPED Context1 IV1 (IPEDCTX1IV1)	32	RW	0000_0000h
548h	Start Address of Region (IPEDCTX1START)	32	RW	0000_0000h
54Ch	End Address of Region (IPEDCTX1END)	32	RW	0000_0000h
560h	IPED Context2 IV0 (IPEDCTX2IV0)	32	RW	0000_0000h
564h	IPED Context2 IV1 (IPEDCTX2IV1)	32	RW	0000_0000h
568h	Start Address of Region (IPEDCTX2START)	32	RW	0000_0000h
56Ch	End Address of Region (IPEDCTX2END)	32	RW	0000_0000h
580h	IPED Context3 IV0 (IPEDCTX3IV0)	32	RW	0000_0000h
584h	IPED Context3 IV1 (IPEDCTX3IV1)	32	RW	0000_0000h
588h	Start Address of Region (IPEDCTX3START)	32	RW	0000_0000h
58Ch	End Address of Region (IPEDCTX3END)	32	RW	0000_0000h
5A0h	IPED Context4 IV0 (IPEDCTX4IV0)	32	RW	0000_0000h
5A4h	IPED Context4 IV1 (IPEDCTX4IV1)	32	RW	0000_0000h
5A8h	Start Address of Region (IPEDCTX4START)	32	RW	0000_0000h
5ACh	End Address of Region (IPEDCTX4END)	32	RW	0000_0000h
5C0h	IPED Context5 IV0 (IPEDCTX5IV0)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
5C4h	IPED Context5 IV1 (IPEDCTX5IV1)	32	RW	0000_0000h
5C8h	Start Address of Region (IPEDCTX5START)	32	RW	0000_0000h
5CCh	End Address of Region (IPEDCTX5END)	32	RW	0000_0000h
5E0h	IPED Context6 IV0 (IPEDCTX6IV0)	32	RW	0000_0000h
5E4h	IPED Context6 IV1 (IPEDCTX6IV1)	32	RW	0000_0000h
5E8h	Start Address of Region (IPEDCTX6START)	32	RW	0000_0000h
5ECh	End Address of Region (IPEDCTX6END)	32	RW	0000_0000h

10.7.2.2 Module Control 0 (MCR0)

Offset

Register	Offset
MCR0	0h

Function

Controls basic functions of FlexSPI module

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AHBGRANTWAIT								IPGRANTWAIT							
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LEARNEN	SCKFREE...	COMBINA...	DOZEEN	HSEN	SERCLKDIV			0	0	RXCLKSRC		Reserved		MDIS	SWRESET
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-24 AHBGRANTWAIT	Timeouts Wait Cycle for AHB command Grant Sets duration of timeout wait cycle for AHB commands. If the arbitrator does not grant the AHB-triggered command, it times out after (AHBGRANTTIMEOUT × 1024) AHB clock cycles. When the pending

Table continues on the next page...

Table continued from the previous page...

Field	Function
	command sequence is IP-triggered and the read or write data size is too large, this grant timeout may occur. When an AHB command grant timeout occurs, an INTR[AHBCMDGE] interrupt is generated. When INTEN[AHBCMDGEEN] = 1, the arbitrator ignores AHB commands. **NOTE** This field is for debugging only. Do not change the value from its default.
23-16 IPGRANTWAIT	Timeout Wait Cycle for IP Command Grant Sets duration of timeout wait cycle for IP commands. If the arbitrator does not grant the IP-triggered command, it times out after (IPGRANTTIMEOUT × 1024) AHB clock cycles. When the pending command sequence is AHB-triggered and the read or write data size is too large, this grant timeout may occur. When IP command grant timeout occurs, an INTR[IPCMDGE] interrupt is generated. When INTEN[IPCMDGEEN] = 1, the arbitrator ignores IP commands. **NOTE** This field is for debugging only. Do not change the value from its default.
15 LEARNEN	Data Learning Enable Enables data learning feature. When data learning is disabled, the sampling clock phase 0 is always used for receive data sampling, even when a learn instruction is correctly executed. See Data learning. 0b - Disable 1b - Enable
14 SCKFREERUNEN	SCLK Free-running Enable Enables free-running SCLK output. For FPGA applications, the external device may use SCLK as a reference clock to its internal PLL. If SCLK free-running is enabled, data sampling with loopback clock from SCLK pad is not supported (MCR0[RXCLKSRC] = 2). 0b - Disable 1b - Enable
13 COMBINATIONEN	Combination Mode Enable Enables combination mode, which supports flash Octal mode access. When Port A and Port B data are four bits wide, this mode combines Port A and B Data pins (A_DATA[3:0] and B_DATA[3:0]). When Port A and Port B data are eight bits wide, combination mode is not supported; write 0 to this field in this case. 0b - Disable 1b - Enable
12 DOZEEN	Doze Mode Enable Enables Doze mode. When enabled, AHB clock and serial clock are gated off when there is a Doze-mode request from the system. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 HSEN	Half Speed Serial Flash Memory Access Enable Enables the clock divider to provide a half-speed clock to external serial flash devices (A_SCLK/ B_SCLK) for all commands in SDR and DDR mode. Write 1 to MCR0[MDIS] before changing the value of this field. Failure to do so may cause issues in the internal logic or state machine. 0b - Disable 1b - Enable
10-8 SERCLKDIV	Serial Root Clock Divider Sets divider value for serial root clock. The serial root clock can be divided inside FlexSPI . See Clocking. **NOTE** Do not modify this field while FlexSPI is active (MCR0[MDIS] = 0). 000b - Divided by 1 001b - Divided by 2 010b - Divided by 3 011b - Divided by 4 100b - Divided by 5 101b - Divided by 6 110b - Divided by 7 111b - Divided by 8
7 —	Reserved
6 —	Reserved
5-4 RXCLKSRC	Sample Clock Source for Flash Reading Selects sampling clock source for flash memory reading. See Receive clock source features. 00b - Dummy Read strobe that FlexSPI generates, looped back internally 01b - Dummy Read strobe that FlexSPI generates, looped back from DQS pad 10b - SCLK output clock and looped back from SCLK pad 11b - Flash-memory-provided read strobe and input from DQS pad
3-2 —	Reserved
1	Module Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
MDIS	Disables FlexSPI module. When the module is disabled, AHB and serial clock are gated off internally to save power. Only register access is allowed, except for the LUT, IP receive FIFO, and IP transmit FIFO. 0b - No impact 1b - Module disable
0 SWRESET	Software Reset Resets the internal FlexSPI state machine and aborts any transactions in progress. Hardware automatically writes 0 to this field after software reset is done. Configuration registers are not reset. 0b - No impact 1b - Software reset

10.7.2.3 Module Control 1 (MCR1)

Offset

Register	Offset
MCR1	4h

Function

Controls basic functions of FlexSPI module

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SEQWAIT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AHBBUSWAIT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-16	Command Sequence Wait

Table continues on the next page...

Table continued from the previous page...

Field	Function
SEQWAIT	Sets wait time for command sequence. Command sequence execution times out and aborts after (SEQWAIT × 1024) serial root clock cycles. When this timeout occurs, if the interrupt is enabled (INTEN[SEQTIMEOUTEN] = 1), an INTR[SEQTIMEOUT] interrupt is generated. Also, the arbitrator ignores AHB commands. **NOTE** You cannot write 0 to this field.
15-0 AHBBUSWAIT	AHB Bus Wait Sets wait time for AHB bus. When data is not received from or transmitted to serial flash memory space after (AHBBUSWAIT × 1024) AHB clock cycles, the read or write access times out. When this timeout occurs, the AHB bus receives an error response, and an interrupt is generated (INTR[AHBBUSTIMEOUT]). When INTR[AHBBUSTIMEOUT] = 1, the arbitrator ignores AHB commands. **NOTE** You cannot write 0 to this field.

10.7.2.4 Module Control 2 (MCR2)

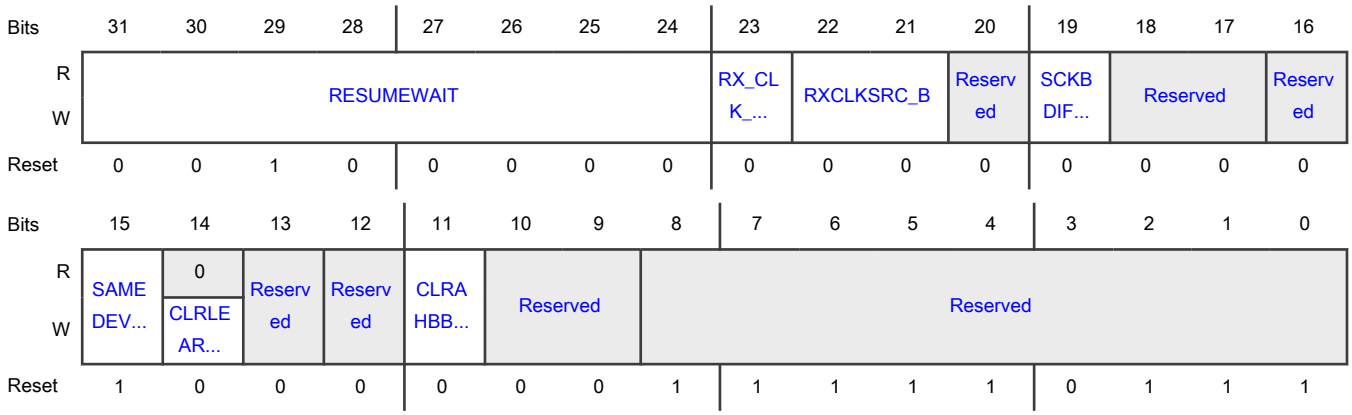
Offset

Register	Offset
MCR2	8h

Function

Controls basic functions of FlexSPI module.

Diagram



Fields

Field	Function
31-24 RESUMEWAIT	Resume Wait Duration Determines duration (in AHB clock cycles) to remain in idle state before suspended command sequence is resumed. See Command abort and suspend.
23 RX_CLK_SRC_DIFF	Sample Clock Source Different Selects whether to use the same clock source or a different clock source for Port A and Port B. 0b - Use MCR0[RXCLKSRC] for Port A and Port B. MCR2[RXCLKSRC_B] is ignored and MCR0[RXCLKSRC] selects the Sample Clock source for Flash Reading of both ports A and B. 1b - Use MCR0[RXCLKSRC] for Port A, and MCR2[RXCLKSRC_B] for Port B. MCR0[RXCLKSRC] selects the Sample Clock source for Flash Reading of port A (A_SCLK) and MCR2[RXCLKSRC_B] selects the Sample Clock source for Flash Reading of port B (B_SCLK).
22-21 RXCLKSRC_B	Port B Receiver Clock Source Selects Port B sample clock source selection for flash memory reading. 00b - Dummy read strobe that FlexSPI generates, looped back internally. 01b - Dummy read strobe that FlexSPI generates, looped back from DQS pad. 10b - SCLK output clock and looped back from SCLK pad 11b - Flash-memory-provided read strobe and input from DQS pad
20 —	Reserved
19 SCKBDIFFOPT	SCLK Port B Differential Output Controls whether to use B_SCLK pad as the Port B SCLK output. Alternatively, B_SCLK pad can be used as A_SCLK differential clock output (inverted clock to A_SCLK). Before changing the value of this field, write 1 to MCR0[MDIS]. After changing the value of this field, write 1 to MCR0[SWRESET]. 0b - Use B_SCLK pad as port B SCLK clock output. Port B flash memory access is available. 1b - Use B_SCLK pad as port A SCLK inverted clock output (Differential clock to A_SCLK). Port B flash memory access is not available.
18-17 —	Reserved
16 —	Reserved
15 SAMEDEVICEEN	Same Device Enable Sets all external devices (A1, A2, B1, and B2) to be the same devices (in type and in size).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - In Individual mode, FLSHA1CRx and FLSHA2CRx, FLSHB1CRx and FLSHB2CRx settings are applied to Flash A1, A2, B1, B2 separately. In Parallel mode, FLSHA1CRx register setting is applied to Flash A1 and B1, FLSHA2CRx register setting is applied to Flash A2 and B2. FLSHB1CRx and FLSHB2CRx register settings are ignored. 1b - FLSHA1CR0, FLSHA1CR1, and FLSHA1CR2 register settings are applied to Flash A1, A2, B1, B2. FLSHA2CRx, FLSHB1CRx, and FLSHB2CRx settings are ignored.
14 CLRLEARNPHASE	Clear Learn Phase Selection Resets the sampling clock phase selection to 0. When 1 is written to this field, it becomes 0 again immediately. 0b - No impact 1b - Reset sample clock phase selection to 0
13 —	Reserved
12 —	Reserved
11 CLRAHBBUFOPT	Clear AHB Buffer Determines whether AHB receive and transmit buffers are cleared automatically when FlexSPI returns Stop mode ACK. If an AHB receive buffer or transmit buffer will be powered off in Stop mode, software should write 1 to this field. Otherwise, an AHB read access after exiting Stop mode may hit either buffer, but their data entries are invalid. 0b - Not cleared automatically 1b - Cleared automatically
10-9 —	Reserved
8-0 —	Reserved

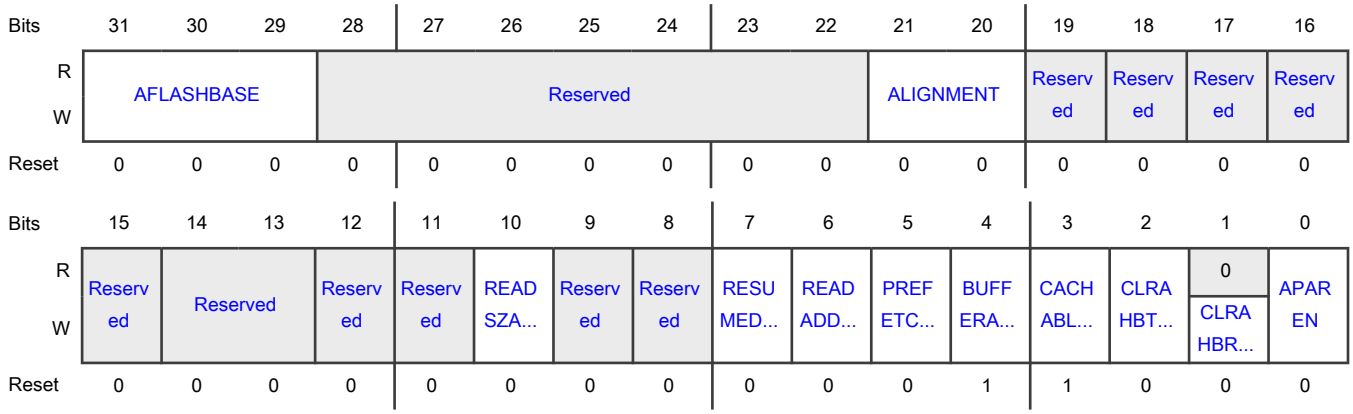
10.7.2.5 AHB Bus Control (AHBCR)

Offset

Register	Offset
AHBCR	Ch

Function
Controls AHB bus interface functions.

Diagram



Fields

Field	Function
31-29 AFLASHBASE	AHB Memory-Mapped Flash Base Address
28-22 —	Reserved
21-20 ALIGNMENT	AHB Boundary Alignment Configures the size of AHB read and write boundary. All accesses that cross the boundary are divided into smaller sub accesses. 00b - No limit 01b - 1 KB 10b - 512 bytes 11b - 256 bytes
19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
14-13 —	Reserved
12 —	Reserved
11 —	Reserved
10 READSZALIGN	AHB Read Size Alignment Configures how AHB read size is determined. See Table 84. 0b - Register settings such as PREFETCH_EN determine AHB read size. 1b - AHB read size to up size to 8 bytes aligned, no prefetching
9 —	Reserved
8 —	Reserved
7 RESUMEDISABLE	AHB Read Resume Disable Disables the resumption of AHB read prefetch. 0b - Suspended AHB read prefetch resumes when AHB is IDLE. 1b - Suspended AHB read prefetch does not resume once aborted.
6 READADDROPT	AHB Read Address Option Removes AHB burst start address alignment limitation. When the FlexSPI controller is used for FPGA application, this field may be required for FlexSPI to fetch the exact byte number as an AHB burst. In this case, FPGA device should be non-word-addressable and this field must be 0. 0b - AHB read burst start address alignment is limited when flash memory is accessed in parallel mode or flash is word-addressable. 1b - AHB read burst start address alignment is not limited. FlexSPI fetches more data than the AHB burst requires for address alignment.
5 PREFETCHEN	AHB Read Prefetch Enable Enables AHB read prefetch. When enabled, FlexSPI fetches more flash read data than the current AHB burst requires. This action reduces the read latency for next AHB read access. **NOTE** AHB read prefetch is enabled only when both this field and AHBRXBUFnCR0[PREFETCHEN] are 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
4 BUFFERABLEEN	Bufferable Write Access Enable Enables AHB bus bufferable write access. This field affects the last beat of an AHB write access. See AHB write access to flash memory. 0b - Disabled. For all AHB write accesses (bufferable or nonbufferable), FlexSPI returns AHB Bus Ready after transmitting all data and finishing command. 1b - Enabled. For AHB bufferable write access, FlexSPI returns AHB Bus Ready when the arbitrator grants the AHB command. FlexSPI does not wait for the AHB command to finish.
3 CACHABLEEN	Cacheable Read Access Enable Enables AHB bus cacheable read access. 0b - Disabled. When an AHB bus cacheable read access occurs, FlexSPI does not check whether it hit the AHB transmit buffer. 1b - Enabled. When an AHB bus cacheable read access occurs, FlexSPI first checks whether the access hit the AHB transmit buffer.
2 CLRAHBTXBUF	Clear AHB Transmit Buffer Clears status and pointers of AHB transmit buffer. When the clear operation completes, this field clears automatically, so it always reads as 0. 0b - No impact. 1b - Enable clear operation.
1 CLRAHBRXBUF	Clear AHB Receive Buffer Clears status and pointers of AHB receive buffer. When the clear operation completes, this field clears automatically, so it always reads as 0. 0b - No impact. 1b - Enable clear operation.
0 APAREN	AHB Parallel Mode Enable Enables Parallel mode for AHB-triggered read and write commands. 0b - Flash is accessed in Individual mode. 1b - Flash is accessed in Parallel mode.

10.7.2.6 Interrupt Enable (INTEN)

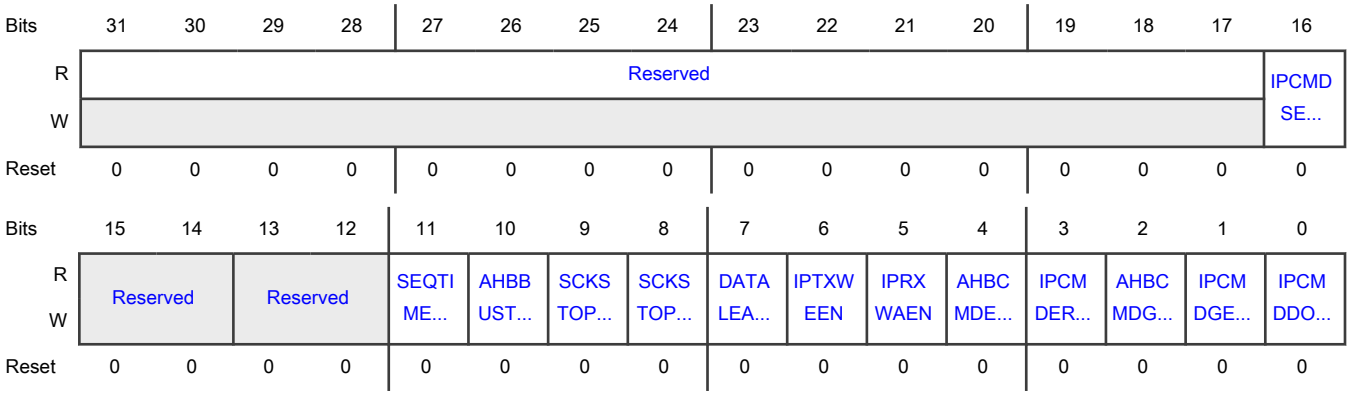
Offset

Register	Offset
INTEN	10h

Function

Includes AHB error response, IPS error response, and ECC error interrupt enables. See Interrupts.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 IPCMDSECURVIOEN	IP Command Security Violation Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUTEN	Sequence execution Timeout Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt
10 AHBBUSTIMEOUTEN	AHB Bus Timeout Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt
9 SCKSTOPBYWREN	SCLK Stopped By Write Interrupt Enable Enables interrupt that indicates SCLK is stopped during command sequence because asynchronous transmit FIFO is empty. 0b - Disable interrupt or no impact 1b - Enable interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 SCKSTOPBYRDEN	SCLK Stopped By Read Interrupt Enable Enables interrupt that indicates SCLK is stopped during command sequence because asynchronous receive FIFO is full. 0b - Disable interrupt or no impact 1b - Enable interrupt
7 DATALEARNFAILEN	Data Learning Failed Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt
6 IPTXWEEN	IP Transmit FIFO Watermark Empty Interrupt Enable Enables interrupt that indicates IP transmit FIFO contains more empty space than watermark level. 0b - Disable interrupt or no impact 1b - Enable interrupt
5 IPRXWAEN	IP Receive FIFO Watermark Available Interrupt Enable Enables interrupt that indicates IP receive FIFO contains more valid data than the watermark level. 0b - Disable interrupt or no impact 1b - Enable interrupt
4 AHBCMDERREN	AHB-Triggered Command Sequences Error Detected Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt
3 IPCMDERREN	IP-Triggered Command Sequences Error Detected Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt
2 AHBCMDGEEN	AHB-Triggered Command Sequences Grant Timeout Interrupt Enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
1 IPCMDGEEN	IP-Triggered Command Sequences Grant Timeout Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt
0 IPCMDDONEEN	IP-Triggered Command Sequences Execution Finished Interrupt Enable 0b - Disable interrupt or no impact 1b - Enable interrupt

10.7.2.7 Interrupt (INTR)

Offset

Register	Offset
INTR	14h

Function

Includes AHB error response, IPS error response, and ECC error interrupts. See Interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															IPCMDSE...
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Reserved		SEQTIME...	AHBBUST...	SCKSTOP...	SCKSTOP...	DATALEA...	IPTXWE	IPRXWA	AHBCMDE...	IPCMDERR	AHBCMDGE	IPCMDGE	IPCMDDO...
W					W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 IPCMDSECUREVIO	IP Command Security Violation **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
15-14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 —	Reserved
11 SEQTIMEOUT	Sequence Execution Timeout **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
10 AHBBUSTIMEOUT	AHB Bus Timeout **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
9 SCKSTOPBYWR	SCLK Stopped Due To Empty Transmit FIFO Generated when SCLK is stopped during command sequence because the asynchronous transmit FIFO is empty. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
8	SCLK Stopped Due To Full Receive FIFO

Table continues on the next page...

Table continued from the previous page...

Field	Function
SCKSTOPBYRD	Generated when SCLK is stopped during command sequence because the asynchronous receive FIFO is full. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
7 DATALEARNFAIL	Data Learning Failed **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
6 IPTXWE	IP Transmit FIFO Watermark Empty Generated when the IP transmit FIFO contains more empty space than the watermark level. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
5 IPRXWA	IP Receive FIFO Watermark Available Generated when the IP receive FIFO contains more valid data than the watermark level.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
4 AHBCMDERR	AHB-Triggered Command Sequences Error Generated upon an AHB-triggered command sequence error. When an error is detected for an AHB command, the command is ignored and not executed. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
3 IPCMDERR	IP-Triggered Command Sequences Error Generated upon an IP-triggered command sequence error. When an error is detected for an IP command, the command is ignored and not executed. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
2 AHBCMDGE	AHB-Triggered Command Sequences Grant Timeout **NOTE** This field behaves differently for register reads and writes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
1 IPCMDGE	IP-Triggered Command Sequences Grant Timeout **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag
0 IPCMDDONE	IP-Triggered Command Sequences Execution Finished Generated upon completion of an IP-triggered command sequence. Also generated when IPCMDGE or IPCMDERR interrupt is generated **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt condition has not occurred 1b - Interrupt condition has occurred When writing 0b - No effect 1b - Clear the flag

10.7.2.8 LUT Key (LUTKEY)

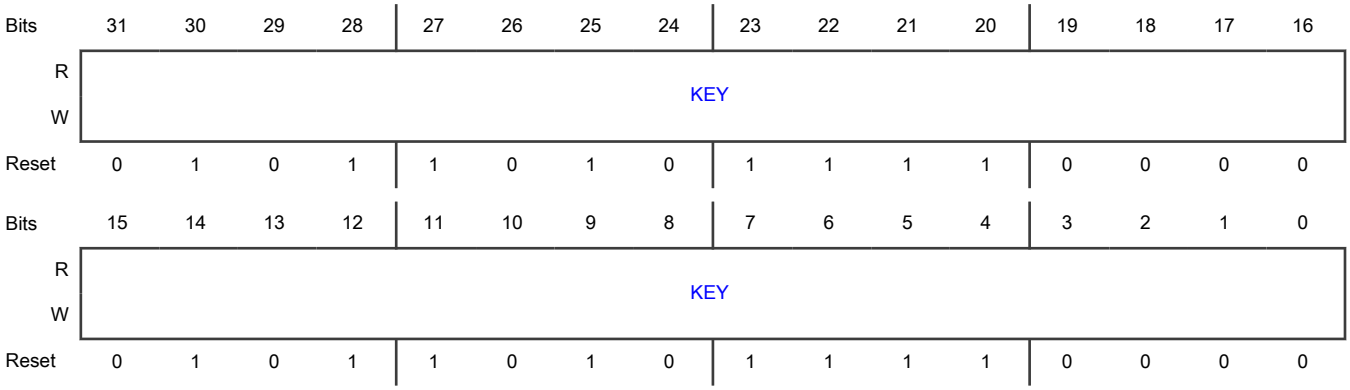
Offset

Register	Offset
LUTKEY	18h

Function

Contains the key to lock and unlock LUT. See Lookup table (LUT).

Diagram



Fields

Field	Function
31-0 KEY	LUT Key Contains key to lock or unlock LUT. The key is 5AF05AF0h. Read value is always 5AF05AF0h.

10.7.2.9 LUT Control (LUTCR)

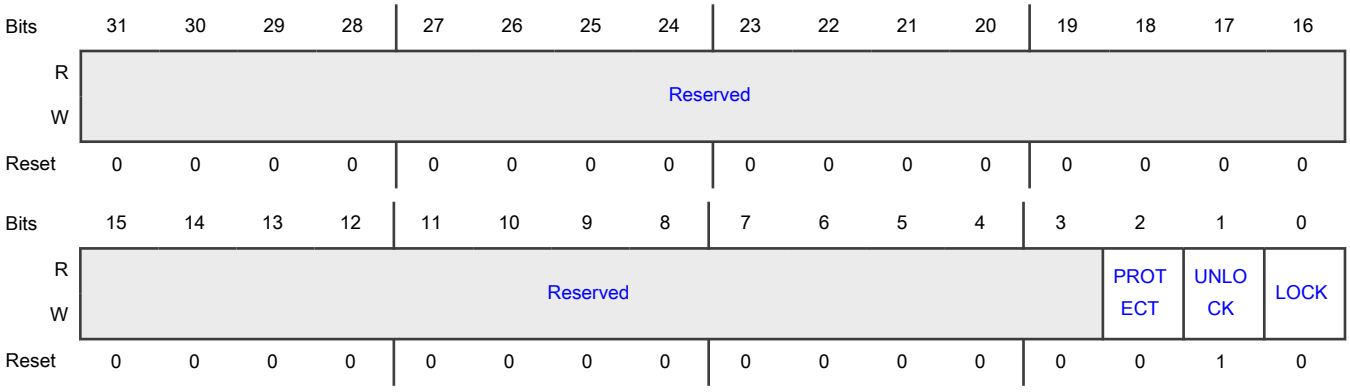
Offset

Register	Offset
LUTCR	1Ch

Function

Used with LUTKEY register to lock or unlock LUT. For the lock or unlock operation to be successful, this register must be written immediately after writing 5AF05AF0h to the LUTKEY register. See Lookup table (LUT) for details on locking and unlocking LUT. You cannot write 00 or 11 to the LOCK and UNLOCK fields.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 PROTECT	LUT Protection Controls lookup table protection. 0b - Not protected. All IPS controllers can access LUTCR and LUT memory. 1b - Protected. Only secure IPS controller can change the value of LUTCR and write to LUT memory.
1 UNLOCK	Unlock LUT 0b - LUT is locked (LUTCR[LOCK] must be 1) 1b - LUT is unlocked and can be written
0 LOCK	Lock LUT 0b - LUT is unlocked (LUTCR[UNLOCK] must be 1) 1b - LUT is locked and cannot be written

10.7.2.10 AHB Receive Buffer n Control 0 (AHBRXBUF0CR0 - AHBRXBUF7CR0)

Offset

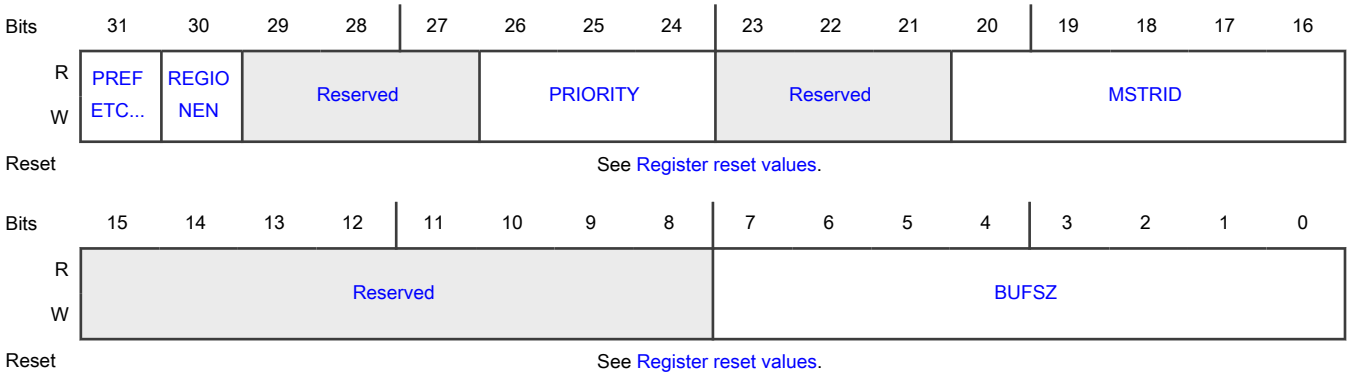
For n = 0 to 7:

Register	Offset
AHBRXBUFnCR0	20h + (n × 4h)

Function

Stores the read data from the SPI interface.

Diagram



Register reset values

Register	Reset value
AHBRXBUF0CR0	8000_0010h
AHBRXBUF1CR0	8001_0010h
AHBRXBUF2CR0	8002_0010h
AHBRXBUF3CR0	8003_0010h
AHBRXBUF4CR0	8004_0010h
AHBRXBUF5CR0	8005_0010h
AHBRXBUF6CR0	8006_0010h
AHBRXBUF7CR0	8007_0010h

Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable Enables AHB read prefetch for the controller corresponding to the current AHB receive buffer. The prefetch feature is disabled when AHBCR[PREFETCHEN] is 0. You can use this field to enable or disable prefetch separately for each controller. 0b - Disabled 1b - Enabled when AHBCR[PREFETCHEN] is enabled.
30 REGIONEN	AHB Receive Buffer Address Region Enable Enable prefetch buffer enhancement. See Prefetch buffer enhancement. **NOTE** Only AHBRXBUF 0-3 have this function. 0b - Disabled. The buffer hit is based on the value of MSTRID only. 1b - Enabled. The buffer hit is based on the value of MSTRID and the address within AHBBUFREGIONSTARTn and AHBREGIONENDn.
29-27 —	Reserved
26-24 PRIORITY	AHB Controller Read Priority Configure the priority for the AHB Controller Read to which this AHB receive buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend.
23-21 —	Reserved
20-16	AHB Controller ID

Table continues on the next page...

Table continued from the previous page...

Field	Function
MSTRID	Configures the ID of the AHB controller to which this AHB receive buffer is assigned. See AHB receive buffer management.
15-8 —	Reserved
7-0 BUFSZ	AHB Receive Buffer Size Configures the size of the AHB receive buffer in multiples of 64 bits. See AHB receive buffer management.

10.7.2.11 Flash Control 0 (FLSHA1CR0 - FLSHB2CR0)

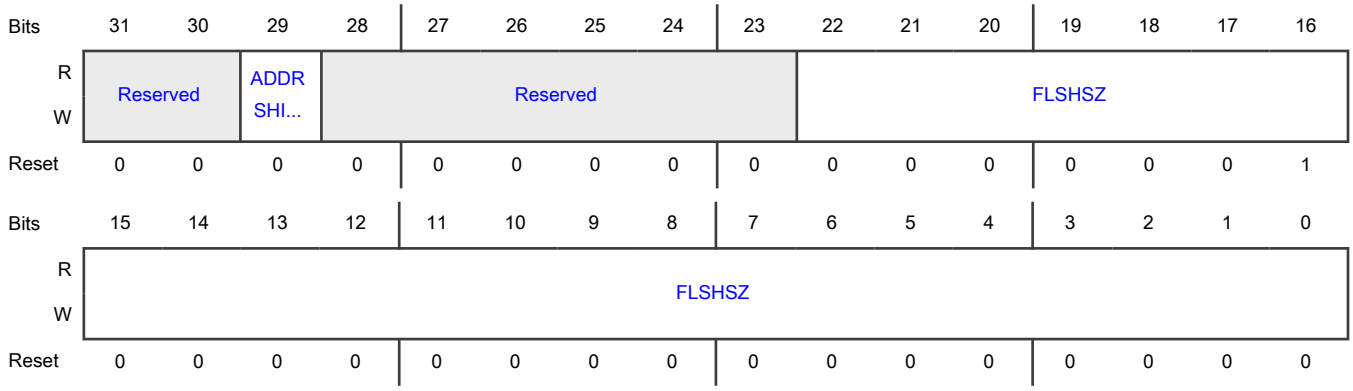
Offset

Register	Offset
FLSHA1CR0	60h
FLSHA2CR0	64h
FLSHB1CR0	68h
FLSHB2CR0	6Ch

Function

Contains flash memory size setting. FlexSPI determines which device is accessed (Chip Select) via this register.

Diagram



Fields

Field	Function
31-30	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
29 ADDRSHIFT	AHB Address Shift Function control Used to left-shift five bits for the flash address to support ISSI quad PSRAM. 0b - Disabled 1b - Enabled
28-23 —	Reserved
22-0 FLSHSZ	Flash Size in KB Configures the maximum flash memory size. For AHB access: Maximum flash size supported for each device is 512 MB. When the value of this field is greater than the maximum size, the device flash size is interpreted as the maximum. The maximum flash size is the largest flash size supported for a single device. It is also the maximum total flash size supported for all devices (up to four). If the total flash size is larger than the maximum size, only the maximum flash size address space is accessible. For IPS access, The maximum flash size supported for each device is 4 GB. When the value of this field is greater than 400000h, the device flash size is taken as 4 GB. The max total flash size supported (for all 4 devices) is also 4 GB. If the total flash size is larger than 4 GB, only 4 GB of address space is accessible.

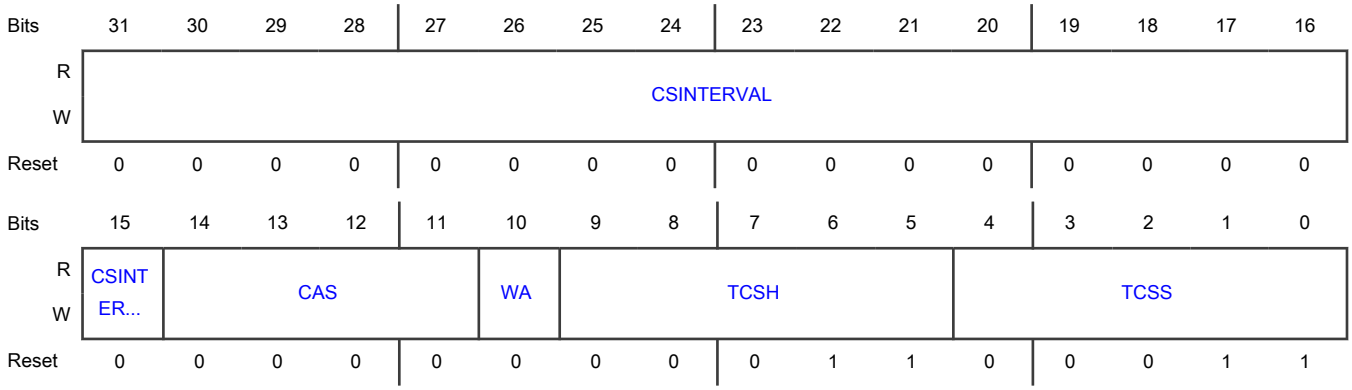
10.7.2.12 Flash Control 1 (FLSHA1CR1 - FLSHB2CR1)

Offset

Register	Offset
FLSHA1CR1	70h
FLSHA2CR1	74h
FLSHB1CR1	78h
FLSHB2CR1	7Ch

Function
Contains settings for flash device-specific timings and flash internal address space.

Diagram



Fields

Field	Function
31-16 CSINTERVAL	Chip Select Interval Configures the minimum interval between flash device chip select deassertion and chip select assertion. If the external flash device has a limitation on the interval between command sequences, configure this field accordingly. If there is no limitation, write 0h to this field. When CSINTERVALUNIT = 0, the chip select invalid interval is: CSINTERVAL × 1 serial clock cycle; When CSINTERVALUNIT = 1, the chip select invalid interval is: CSINTERVAL × 256 serial clock cycles. **NOTE** The minimum chip select interval is 2 cycles, even when the value of CSINTERVAL is less than 2.
15 CSINTERVALUNIT	Chip Select Interval Unit Configures the interval unit for chip select. 0b - 1 serial clock cycle 1b - 256 serial clock cycles
14-11 CAS	Column Address Size When external flash memory has a separate address field for rows and columns, this field configures the flash column address bit width. FlexSPI automatically splits a flash-mapped address into row address and column address according to the values of this field and the WA field. When the external flash memory does not support column address, write 0 to this field. FlexSPI transmits all flash address bits as Row address. For flash address mapping, see Flash address sent to flash memory devices.
10 WA	Word-Addressable Configures whether external flash memory is word-addressable or byte-addressable. If flash memory is word-addressable, it should be accessed in multiples of 16 bits. Currently, FlexSPI does not transmit flash address bit 0 to external flash memory. For flash address mapping, see Flash address sent to flash memory devices.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Byte-addressable 1b - Word-addressable
9-5 TCSH	Serial Flash CS Hold Time Used to meet flash TCSH timing requirement. Serial flash CS Hold time that FlexSPI promises is: (TCSH + 1/2) serial clock cycles for DDR mode, and TCSH serial clock cycles for SDR mode. See Output timing between chip select and SCLK.
4-0 TCSS	Serial Flash CS Setup Time Used to meet flash TCSS timing requirement. Serial flash CS Setup time that FlexSPI promises is: (TCSS + 1/2) serial root clock cycles for SDR and DDR mode. See Output timing between chip select and SCLK.

10.7.2.13 Flash Control 2 (FLSHA1CR2 - FLSHB2CR2)

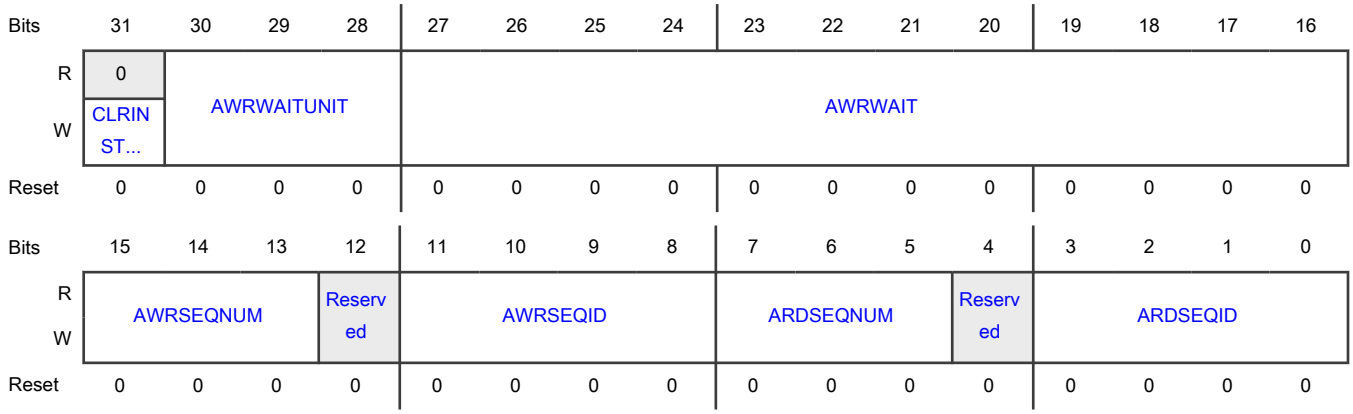
Offset

Register	Offset
FLSHA1CR2	80h
FLSHA2CR2	84h
FLSHB1CR2	88h
FLSHB2CR2	8Ch

Function

Contains fields to configure AHB bus access. If the four external devices are different types, AHB read and write commands may use different command sequences; AHB bus ready wait time may also differ.

Diagram



Fields

Field	Function
31 CLRINSTRPTR	Clear Instruction Pointer Clears the instruction pointer, which is the pointer that JMP_ON_CS saves internally. See Programmable sequence engine. This field is used for AHB read access to external flash memory supporting Execute-In-Place (XIP) mode.
30-28 AWRWAITUNIT	AWRWAIT Unit Configures the unit of AHB write wait time, as the value of AWRWAIT determines, in terms of AHB clock cycles. 000b - 2 001b - 8 010b - 32 011b - 128 100b - 512 101b - 2048 110b - 8192 111b - 32768
27-16 AWRWAIT	AHB Write Wait Configures AHB write wait time, with the AWRWAITUNIT field. Certain devices (such as FPGA) require time to write data into internal memory after the command sequences finished on the FlexSPI interface. If another read command sequence arrives before the current programming finishes internally, the read data may be wrong. This field is used to hold the AHB bus ready for AHB write access, waiting until the programming is finished in the external device. This hold ensures that no AHB read command is triggered before the programming finishes in the external device. The wait cycle between an AHB-triggered command sequence finishing on FlexSPI and the AHB return bus being ready is: AWRWAIT × AWRWAITUNIT.
15-13 AWRSEQNUM	Sequence Number for AHB Write-Triggered Command Configures the sequence number of an AHB read-triggered command. For certain flash devices (for example, HyperFlash, HyperRam, and Serial NAND flash), a flash programming access is done via several command sequences. An AHB write command triggers (AWRSEQNUM + 1) command sequences to external flash memory each time. FlexSPI executes the sequences in LUT incrementally. **NOTE** • Software should ensure that the last sequence index never exceeds LUT sequence numbers: AWRSEQID + AWRSEQNUM < 16 • Software must ensure that the AWRSEQNUM and LUT fields are configured correctly according to the external device specification. FlexSPI does not check the sequence; it executes the sequences one by one.
12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 AWRSEQID	Sequence Index for AHB Write-Triggered Command
7-5 ARDSEQNUM	Sequence Number for AHB Read-Triggered Command Configures the sequence number of an AHB read-triggered command in the lookup table. For certain flash devices (for example, HyperFlash, HyperRam, and Serial NAND flash), a flash read access is done via several command sequences. An AHB read command triggers (ARDSEQNUM + 1) command sequences to external flash memory each time. FlexSPI executes the sequences in LUT incrementally. **NOTE** • Software should ensure that the last sequence index never exceeds LUT sequence numbers: ARDSEQID + ARDSEQNUM ≤ 16. • Software must ensure that the ARDSEQNUM and LUT fields are configured correctly according to the external device specification. FlexSPI does not check the sequence; it executes the sequences one by one.
4 —	Reserved
3-0 ARDSEQID	Sequence Index for AHB Read-Triggered Command in LUT

10.7.2.14 Flash Control 4 (FLSHCR4)

Offset

Register	Offset
FLSHCR4	94h

Function

Provides configuration for all external devices.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				Reserved				Reserved		Reserved	Reserved	WMENB	WMENA	WMOPT2	WMOPT1
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

Fields

Field	Function
31-12 —	Reserved
11-9 —	Reserved
8-6 —	Reserved
5 —	Reserved
4 —	Reserved
3 WMENB	Write Mask Enable for Port B Enables write mask for flash device on port B. 0b - Disabled. When writing to external device, DQS(RWDS) pin is not driven. 1b - Enabled. When writing to external device, FlexSPI drives DQS(RWDS) pin as write mask output.
2 WMENA	Write Mask Enable for Port A Enables write mask for flash device on port A. 0b - Disabled. When writing to external device, DQS(RWDS) pin is not driven. 1b - Enabled. When writing to external device, FlexSPI drives DQS(RWDS) pin as write mask output.
1 WMOPT2	Write Mask Option 2 When using AP memory, used to remove AHB or IP write burst minimum length limitation. When using this field, FLSHCR4[WMOPT1] should be 1. 0b - When writing to an external device, DQS pin is used as write mask. When flash memory is accessed in individual mode, AHB or IP write burst length is not limited. 1b - When writing to an external device, DQS pin is not used as write mask. When flash memory is accessed in individual mode, AHB or IP write burst length is limited. The minimum write burst length should be 4.
0 WMOPT1	Write Mask Option 1 Used to remove AHB and IP write burst start address alignment limitation. 0b - When writing to an external device, DQS pin is used as write mask. When flash memory is accessed in individual mode, AHB or IP write burst start address alignment is not limited. 1b - When writing to an external device, DQS pin is not used as write mask. When flash memory is accessed in individual mode, AHB or IP write burst start address alignment is limited.

10.7.2.15 IP Control 0 (IPCR0)

Offset

Register	Offset
IPCR0	A0h

Function

Provides all configuration required for IP commands. Provides the start address for the flash device, instead of the chip address, to be accessed for IP command. FlexSPI determines the chip select automatically according to this start address.

NOTE

- You cannot issue an IP command that crosses flash device boundaries. If you do so, it generates an IPCMDERR interrupt.
- Configure this register before an IP command is triggered.
- Do not change the values in this register while an IP command is in progress.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SFAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SFAR	Serial Flash Address Configures the serial flash address for IP commands. The address should be the address of the flash device without the base address.

10.7.2.16 IP Control 1 (IPCR1)

Offset

Register	Offset
IPCR1	A4h

Function

Provides all configuration required for IP commands. Provides the flash read and program data size, sequence index in LUT, sequence number and individual/parallel mode settings for IP commands.

NOTE

- Configure this register before an IP command is triggered.
- Do not change the values in this register while an IP command is in progress.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IPAREN	Reserved			ISEQNUM				Reserved				ISEQID			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IDATSZ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 IPAREN	Parallel Mode Enable for IP Commands 0b - Disabled. Flash memory is accessed in Individual mode. 1b - Enabled. Flash memory is accessed in Parallel mode.
30-27 —	Reserved
26-24 ISEQNUM	Sequence Number for IP command: ISEQNUM+1.
23-20 —	Reserved
19-16 ISEQID	Sequence Index in LUT for IP command.
15-0 IDATSZ	Flash Read/Program Data Size (in bytes) for IP command.

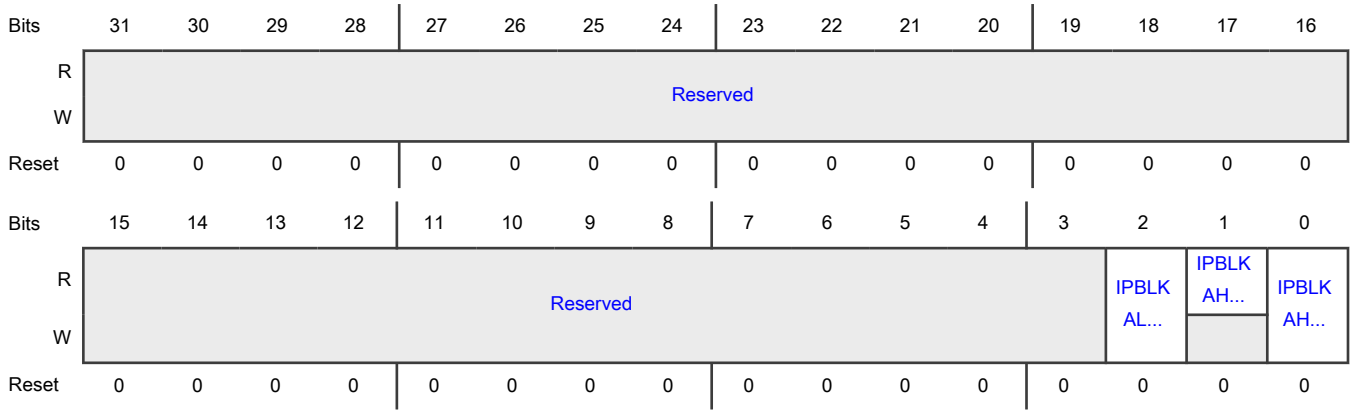
10.7.2.17 IP Control 2 (IPCR2)

Offset

Register	Offset
IPCR2	A8h

Function
Configures whether IP commands can block AHB commands.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 IPBLKALLAHB	IP Command Blocking All AHB Command Enable 0b - IP commands only block AHB commands that affect the IPED region. 1b - IP commands block all AHB commands.
1 IPBLKAHBACK	IP Command Blocking AHB Command Acknowledgment Enable 0b - IP commands do not block AHB command acknowledgment. 1b - IP commands block AHB command acknowledgment.
0 IPBLKAHBREQ	IP Command Blocking AHB Command Request Enable 0b - IP commands do not block AHB command requests. 1b - IP commands block AHB command requests.

10.7.2.18 IP Command (IPCMD)

Offset

Register	Offset
IPCMD	B0h

Function

Used to trigger an IP command to access external flash device. When the arbitrator grants the IP command, the command is executed on the FlexSPI interface.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															0
W																TRG
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 TRG	Command Trigger Triggers an IP command. This field clears automatically after it has been written and always reads as 0. **NOTE** You cannot trigger another IP command before the previous IP command finishes on the FlexSPI interface. Software must poll INTR[IPCMDDONE] or wait for this interrupt. 0b - No action 1b - Start the IP command that the IPCR0 and IPCR1 registers define.

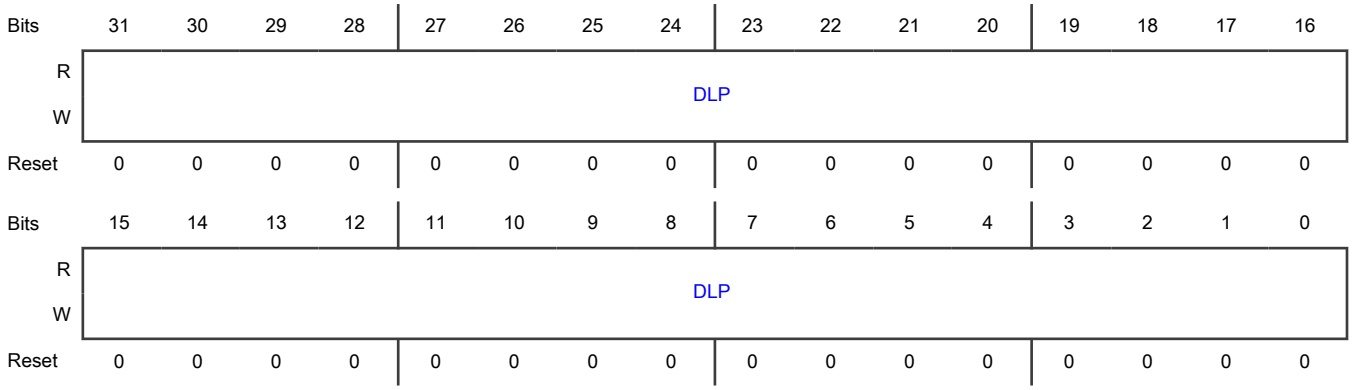
10.7.2.19 Data Learning Pattern (DLPR)

Offset

Register	Offset
DLPR	B4h

Function
Provides the pattern to be used during Data Learning in FlexSPI.

Diagram



Fields

Field	Function
31-0 DLP	Data Learning Pattern The operand in the LEARN_SDR or LEARN_DDR instruction code determines the data learning pattern bit number. This number never exceeds 32 bits. If the operand in the instruction code is greater than 32, a 32-bit pattern is used. See Data learning.

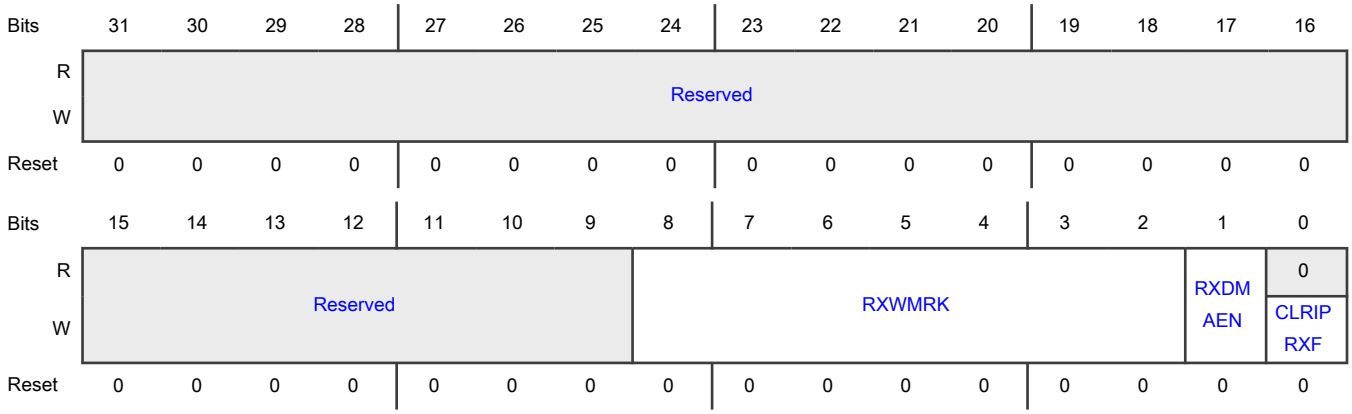
10.7.2.20 IP Receive FIFO Control (IPRXFCR)

Offset

Register	Offset
IPRXFCR	B8h

Function
Provides the configuration fields for IP receive FIFO management.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-2 RXWMRK	IP Receive FIFO Watermark Level Configures the watermark level for the IP receive FIFO. The watermark level is (RXWMRK + 1) × 64 bits. The INTR[IPRXWA] interrupt is set when FlexSPI fills the IP receive FIFO ≥ the watermark level. A DMA request occurs when the fill level is ≥ the watermark level and IPRXFCR[RXDMAEN] = 1. When the fill level is ≥ the watermark level and INTEN[IPRXWAEN] = 1, an INTR[IPRXWA] interrupt is generated. **NOTE** After writing 1 to INTR[IPRXWA] to clear it, the read address should be rolled back to the start address (memory mapped).
1 RXDMAEN	IP Receive FIFO Reading by DMA Enable 0b - Disabled. The processor reads the FIFO. 1b - Enabled. DMA reads the FIFO.
0 CLRIPRXF	Clear IP Receive FIFO Clears all valid data entries in IP receive FIFO. Resets the read and write pointers in IP receive FIFO. 0b - No function 1b - A clock cycle pulse clears all valid data entries in IP receive FIFO.

10.7.2.21 IP Transmit FIFO Control (IPTXFCR)

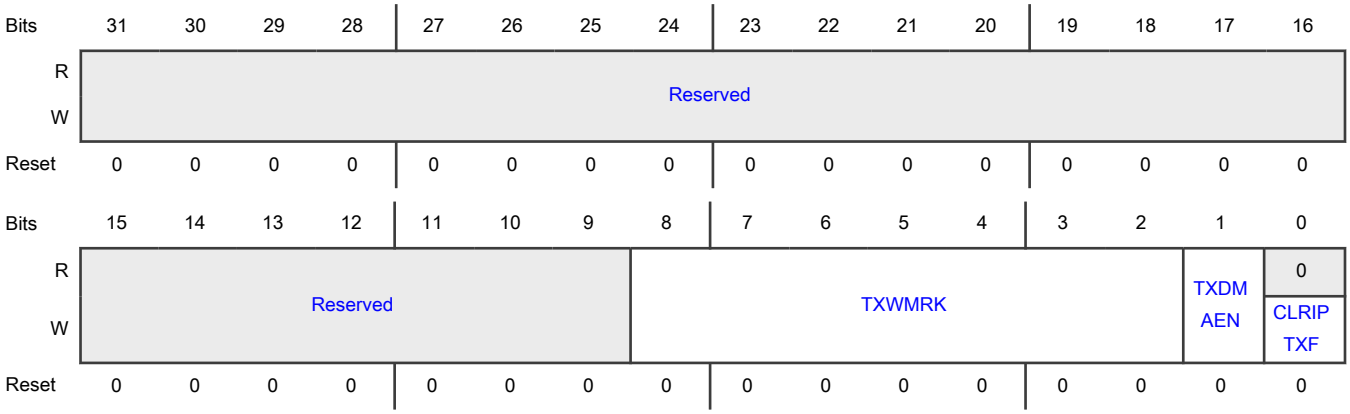
Offset

Register	Offset
IPTXFCR	BCh

Function

Provides the configuration fields for IP transmit FIFO management.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-2 TXWMRK	Transmit Watermark Level Sets the transmit watermark level. The watermark level is (TXWMRK + 1) × 64 bits. INTR[IPTXWE] is set when FlexSPI empties the IP transmit FIFO ≥ the watermark level. When the empty level ≥ the watermark level and IPTXFCR[TXDMAEN] = 1, a DMA request occurs. When the empty level ≥ the watermark level and INTEN[IPTXWEEN] = 1, an INTR[IPTXWE] interrupt is generated. **NOTE** • The watermark level should not be larger than the write window. • The watermark level should not be larger than the IP transmit FIFO size. • The write address to the IP receive FIFO should roll back to the start address of the write window. After pushing the IP transmit fifo, you should write 1 to INTR[IPTXWE] to clear it, which will perform the rollback.
1 TXDMAEN	Transmit FIFO DMA Enable Selects whether DMA or processor fills IP transmit FIFO. 0b - Processor 1b - DMA
0 CLRIPTXF	Clear IP Transmit FIFO Clears all valid data entries in IP transmit FIFO. The read and write pointers in IP transmit FIFO are reset. 0b - No function 1b - A clock cycle pulse clears all valid data entries in the IP transmit FIFO.

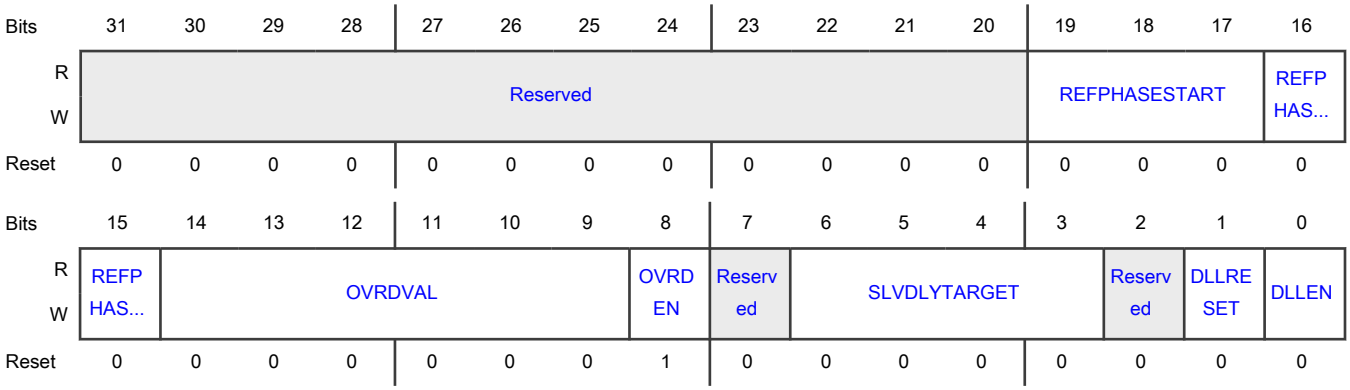
10.7.2.22 DLL Control 0 (DLLACR - DLLBCR)

Offset

Register	Offset
DLLACR	C0h
DLLBCR	C4h

Function
Configures Flash A and B sample clock DLL.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-17 REFPHASESTART	Reference Clock Delay Line Start Phase If the DLL does not lock well when DLLxCR[DLLEN] is enabled, update the register value of REFPHASESTART. 7h is recommended value. Note that DLLxCR[DLLRESET] is required to be triggered after REFPHASESTART is updated.
16-15 REFPHASEGAP	Reference Clock Delay Line Phase Adjust Gap. REFPHASEGAP setting of 2h is recommended if DLLEN is set.
14-9 OVRDVAL	Target Clock Delay Line Override Value Configures the override value for the target clock line delay cell number. When OVRDEN = 1, the delay cell number in DLL is OVRDVAL + 1. See DLL configuration for sampling.
8 OVRDEN	Target Clock Delay Line Override Value Enable Enables the override value for the target clock line delay cell number.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
7 —	Reserved
6-3 SLVDLYTARGET	Target Delay Line Configures the target delay line for the target. The delay target for target delay line is: ((SLVDLYTARGET+1) × 1/32 × clock cycle of reference clock (serial root clock). If serial root clock is ≥ 100 MHz, DLLEN is set to 1, OVRDEN is set to 0, then SLVDLYTARGET setting is up to flash parameters used.
2 —	Reserved
1 DLLRESET	DLL reset Forces a DLL reset. The forced reset causes the DLL to lose lock and recalibrate to detect a ref_clock half-period phase shift. The reset action is edge-triggered, so software must write 0 to this field after writing 1 to it (no delay limitation). 0b - No function 1b - Force DLL reset.
0 DLLEN	DLL Calibration Enable Enables DLL calibration. When DLL calibration is disabled, the delay cell number in the target delay line is always 1. When DLLxCR[OVRDEN] = 1, the target delay line is overridden and this field is ignored. 0b - Disable 1b - Enable

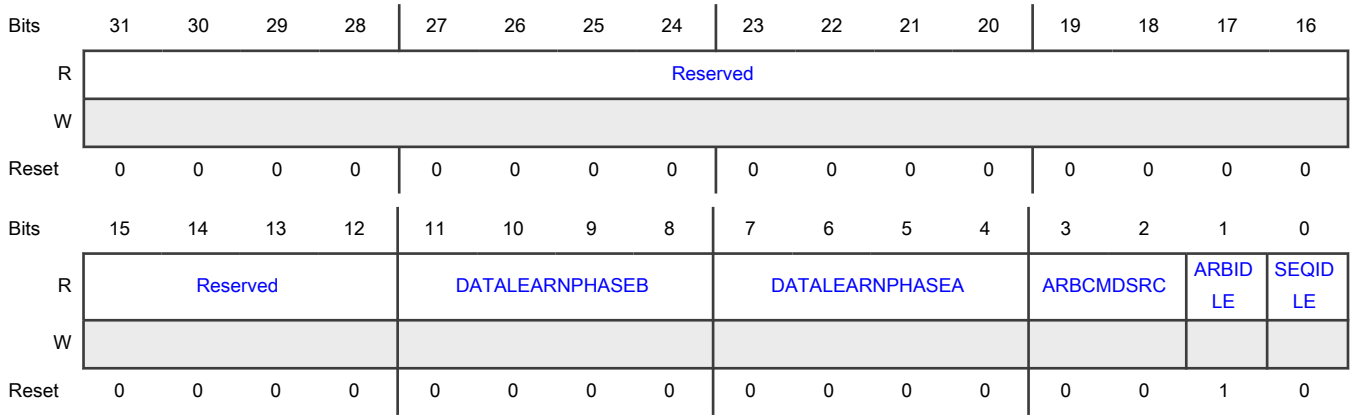
10.7.2.23 Status 0 (STS0)

Offset

Register	Offset
STS0	E0h

Function
Indicates the status of the internal state machine.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-8 DATALEARNPHASEB	Data Learning Phase Selection on Port B Indicates the sampling clock phase selection on Port B after data learning. This field is similar to STS0[DATALEARNPHASEA]. The sampling clock phase is chosen separately for Port A and Port B.
7-4 DATALEARNPHASEA	Data Learning Phase Selection on Port A Indicates the sampling clock phase selection on Port A after data learning. There are 16 clock phases for the sampling clock that the delay cell line generates. When data learning is not enabled, the default clock phase 0 is used to sample flash read data. When data learning is enabled and a LEARN_SDR or LEARN_DDR instruction has executed correctly, FlexSPI determines the correct clock phase to sample read data. See Data learning.
3-2 ARBCMDSRC	ARB Command Source Indicates the trigger source of current command sequence that the arbitrator has granted. When ARB_CTL is not busy (STS0[ARBIDLE] = 1), the value of this field does not matter. 00b - Trigger source is AHB read command. 01b - Trigger source is AHB write command. 10b - Trigger source is IP command (by writing 1 to IPCMD[TRG]). 11b - Trigger source is a suspended command that has resumed.
1 ARBIDLE	ARB_CTL State Machine Idle Indicates whether the state machine in ARB_CTL is idle, or a command sequence that the arbitrator granted has not finished on the FlexSPI interface. When idle, no transaction is occurring on the FlexSPI interface (STS0[SEQIDLE] = 1). When waiting for the FlexSPI controller to become idle, poll this field instead of STS0[SEQIDLE]. 0b - Not idle

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Idle
0 SEQIDLE	SEQ_CTL State Machine Idle Indicates whether the state machine in SEQ_CTL is idle, or a command sequence is executing on the FlexSPI interface. 0b - Not idle 1b - Idle

10.7.2.24 Status 1 (STS1)

Offset

Register	Offset
STS1	E4h

Function
Indicates the error code of the AHB or IPS interface error response.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				IPCMDERRCODE				Reserved				IPCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				AHBCMDERRCODE				Reserved				AHBCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 IPCMDERRCODE	IP Command Error Code When an IP command Error is detected, indicates the error code. This field returns to zero when INTR[IPCMDERR] is cleared.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000b - No error 0010b - IP command with JMP_ON_CS instruction used in the sequence 0011b - Unknown instruction opcode in the sequence 0100b - DUMMY_SDR or DUMMY_RWDS_SDR instruction used in DDR sequence 0101b - DUMMY_DDR or DUMMY_RWDS_DDR instruction used in SDR sequence 0110b - Flash memory access start address exceeds entire flash address range (A1, A2, B1, and B2) 1110b - Sequence execution timeout 1111b - Flash boundary crossed
23-20 —	Reserved
19-16 IPCMDERRID	IP Command Error ID When an IP command error is detected, indicates the sequence index. This field returns to zero when INTR[IPCMDERR] is cleared.
15-12 —	Reserved
11-8 AHBCMDERRCODE	AHB Command Error Code Contains the error code when an AHB command error is detected. This field returns to zero when INTR[AHBCMDERR] is cleared. 0000b - No error 0010b - AHB Write command with JMP_ON_CS instruction used in the sequence 0011b - Unknown instruction opcode in the sequence 0100b - DUMMY_SDR or DUMMY_RWDS_SDR instruction used in DDR sequence 0101b - DUMMY_DDR or DUMMY_RWDS_DDR instruction used in SDR sequence 1110b - Sequence execution timeout
7-4 —	Reserved
3-0 AHBCMDERRID	AHB Command Error ID When an AHB command error is detected, indicates the sequence index. This field returns to zero when INTR[AHBCMDERR] is cleared.

10.7.2.25 Status 2 (STS2)

Offset

Register	Offset
STS2	E8h

Function
Indicates the status of Flash A and B sample clock DLLs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		BREFSEL						BSLVSEL						BREFLOCK	BSLVLOCK
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		AREFSEL						ASLVSEL						AREFLOCK	ASLVLOCK
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29-24 BREFSEL	Flash B Sample Clock Reference Delay Line Delay Cell Number Indicates the sample clock reference delay line delay cell number for Flash B. There are 0-63 phases. • 000001b - Indicates lock phase 0. • 100000b - Indicates lock phase 63.
23-18 BSLVSEL	Flash B Sample Clock Target Delay Line Delay Cell Number Indicates the sample clock target delay line delay cell number for Flash B. There are 0-63 phases. • 000001b - Indicates lock phase 0. • 100000b - Indicates lock phase 63.
17 BREFLOCK	Flash B Sample Clock Reference Delay Line Locked 0b - Not locked 1b - Locked

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 BSLVLOCK	Flash B Sample Target Reference Delay Line Locked 0b - Not locked 1b - Locked
15-14 —	Reserved
13-8 AREFSEL	Flash A Sample Clock Reference Delay Line Delay Cell Number Indicates the sample clock reference delay line delay cell number for Flash A. There are 0-63 phases. • 000001b - Indicates lock phase 0. • 100000b - Indicates lock phase 63.
7-2 ASLVSEL	Flash A Sample Clock Target Delay Line Delay Cell Number Indicates the sample clock target delay line delay cell number for Flash A. There are 0-63 phases. • 000001b - Indicates lock phase 0. • 100000b - Indicates lock phase 63.
1 AREFLOCK	Flash A Sample Clock Reference Delay Line Locked 0b - Not locked 1b - Locked
0 ASLVLOCK	Flash A Sample Target Delay Line Locked 0b - Not locked 1b - Locked

10.7.2.26 AHB Suspend Status (AHBSPNDSTS)

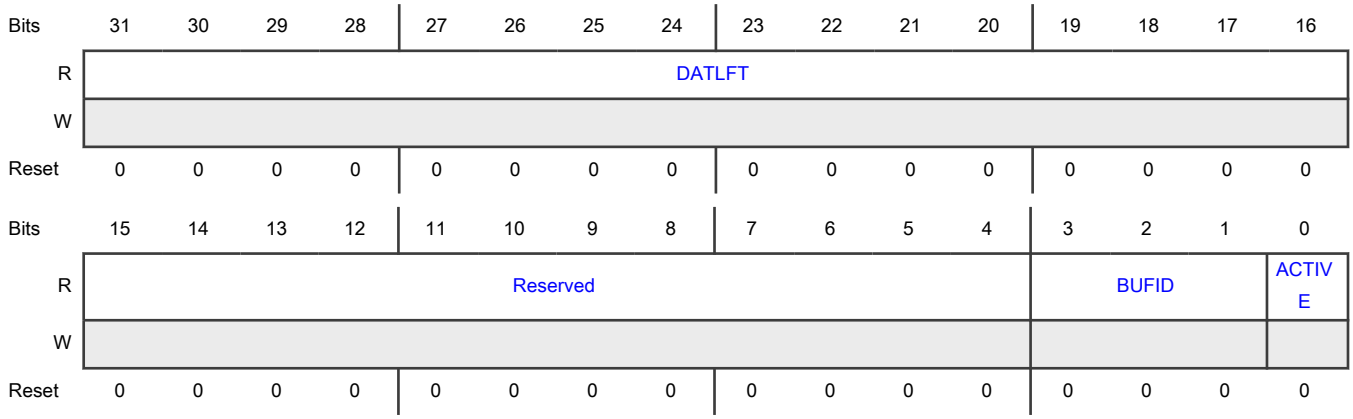
Offset

Register	Offset
AHBSPNDSTS	ECh

Function

Indicates the status of suspended AHB read prefetch command sequence. When an IP or AHB command is triggered while the arbitrator processes an AHB read sequence (prefetching additional data not for current AHB burst), the prefetch sequence is suspended. When there are no longer transactions on FlexSPI, the prefetch sequence may be resumed. FlexSPI saves only one AHB prefetch sequence. When a new prefetch sequence is suspended with an active sequence already suspended, the previous suspended sequence is removed and never resumed. See Command abort and suspend.

Diagram



Fields

Field	Function
31-16 DATLFT	Data Left Contains the data size remaining (in bytes) for a suspended command sequence.
15-4 —	Reserved
3-1 BUFID	AHB Receive Buffer ID for Suspended Command Sequence
0 ACTIVE	Active AHB Read Prefetch Suspended Indicates whether an AHB read prefetch command sequence has been suspended. 0b - No suspended AHB read prefetch command. 1b - An AHB read prefetch command sequence has been suspended.

10.7.2.27 IP Receive FIFO Status (IPRXFSTS)

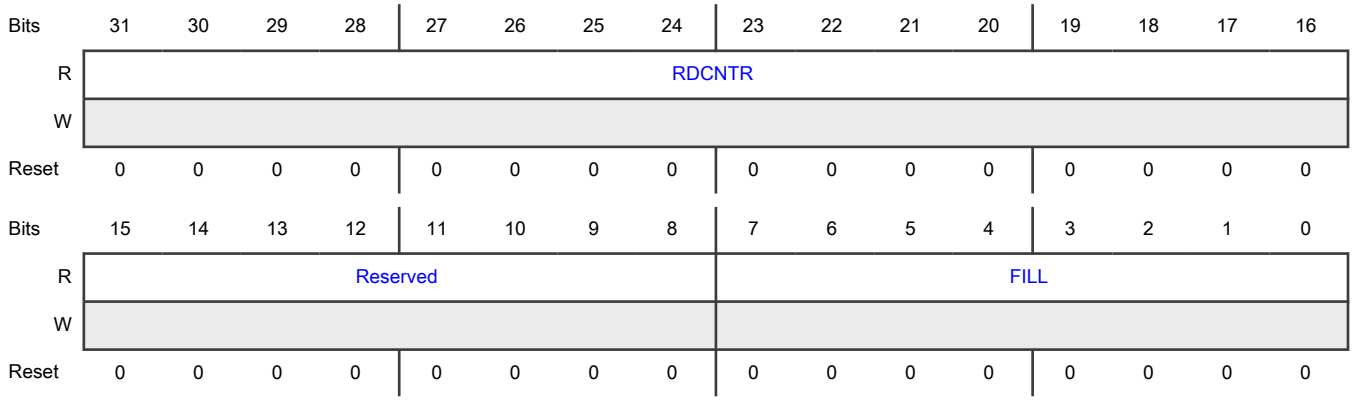
Offset

Register	Offset
IPRXFSTS	F0h

Function

Indicates the status of IP receive FIFO.

Diagram



Fields

Field	Function
31-16 RDCNTR	Read Data Counter Contains counter for total data read. The data read is the value of this field × 64 bits.
15-8 —	Reserved
7-0 FILL	Fill Level of IP Receive FIFO Indicates how full the IP receive FIFO is. The fill level is the value of this field × 64 bits.

10.7.2.28 IP Transmit FIFO Status (IPTXFSTS)

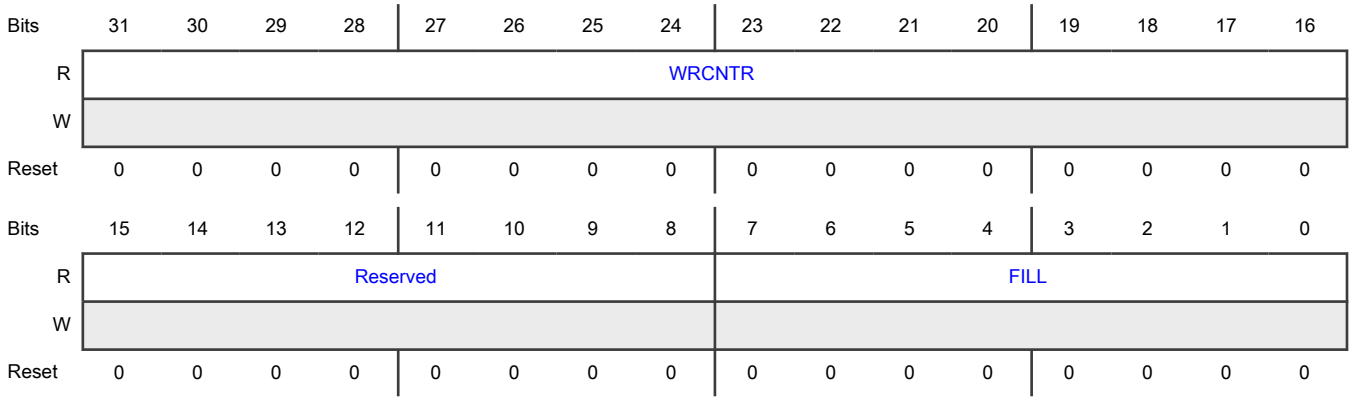
Offset

Register	Offset
IPTXFSTS	F4h

Function

Indicates the status of IP transmit FIFO.

Diagram



Fields

Field	Function
31-16 WRCNTR	Write Data Counter Contains counter for total data written. The data written is the value of this field × 64 bits.
15-8 —	Reserved
7-0 FILL	Fill Level of IP Transmit FIFO Indicates how full the IP transmit FIFO is. The fill level is the value of this field × 64 bits.

10.7.2.29 IP Receive FIFO Data a (RFDR0 - RFDR31)

Offset

For a = 0 to 31:

Register	Offset
RFDRa	100h + (a × 4h)

Function

Provides read access to IP receive FIFO via IPS bus. The read value is unknown for read access to invalid entries in the IP receive FIFO.

NOTE

RFDR is implemented as memory, so the reset value is unknown.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXDATA															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDATA															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 RXDATA	Receive Data Contains receive data. See Reading data from IP receive FIFO.

10.7.2.30 IP TX FIFO Data a (TFDR0 - TFDR31)

Offset

For a = 0 to 31:

Register	Offset
TFDRa	180h + (a × 4h)

Function

Provides write access to IP transmit FIFO via IPS bus.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 TXDATA	Transmit Data Contains transmit data. See Writing data to IP transmit FIFO.

10.7.2.31 Lookup Table a (LUT0 - LUT63)

Offset

For a = 0 to 63:

Register	Offset
LUTa	200h + (a × 4h)

Function

Contains a lookup table for command sequences. Software should set the sequence index before triggering an IP command or AHB command. FlexSPI fetches the command sequence from LUT when an IP or AHB command is triggered. There are 16 command sequences in LUT. See Lookup table (LUT) and Programmable sequence engine.

NOTE

LUT is implemented as memory, so the reset value is unknown.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	OPCODE1						NUM_PADS1		OPERAND1							
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPCODE0						NUM_PADS0		OPERAND0							
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-26 OPCODE1	OPCODE1
25-24 NUM_PADS1	NUM_PADS1

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-16 OPERAND1	OPERAND1
15-10 OPCODE0	OPCODE
9-8 NUM_PADS0	NUM_PADS0
7-0 OPERAND0	OPERAND0

10.7.2.32 HADDR REMAP Start Address (HADDRSTART)

Offset

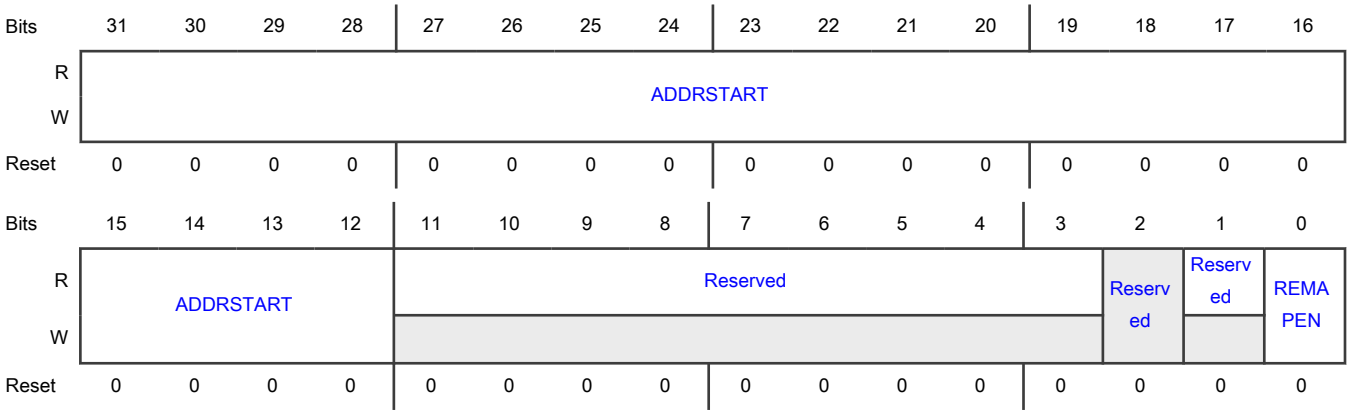
Register	Offset
HADDRSTART	420h

Function

Used to configure options for the dual-image remap feature.

See Dual image use case using HADDRSTART, HADDREND, and HADDROFFSET registers.

Diagram



Fields

Field	Function
31-12	HADDR Start Address

Table continues on the next page...

Table continued from the previous page...

Field	Function
ADDRSTART	Contains the start address of the HADDR remap range, aligned to 4 KB.
11-3 —	Reserved
2 —	Reserved
1 —	Reserved
0 REMAPEN	AHB Bus Address Remap Enable 0b - HADDR REMAP Disabled 1b - HADDR REMAP Enabled

10.7.2.33 HADDR REMAP END ADDR (HADDREND)

Offset

Register	Offset
HADDREND	424h

Function

Used to configure options for the dual-image remap feature.

See Dual image use case using HADDRSTART, HADDREND, and HADDROFFSET registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ENDSTART															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENDSTART				Reserved											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 ENDSTART	End Address of HADDR Remap Range Contains the end address of the HADDR remap range, aligned to 4 KB.
11-0 —	Reserved

10.7.2.34 HADDR Remap Offset (HADDROFFSET)

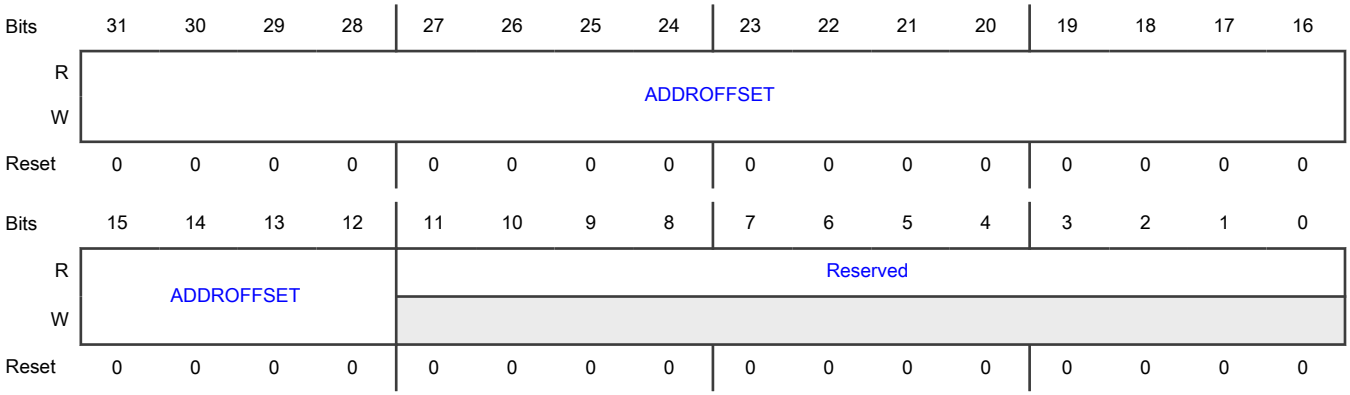
Offset

Register	Offset
HADDROFFSET	428h

Function
Used to configure options for the dual-image remap feature.

See Dual image use case using HADDRSTART, HADDREND, and HADDROFFSET registers.

Diagram



Fields

Field	Function
31-12 ADDROFFSET	HADDR Offset Contains the offset for HADDR. The remapped address is ADDR[31:12] = ADDR_original[31:12] + ADDROFFSET.
11-0 —	Reserved

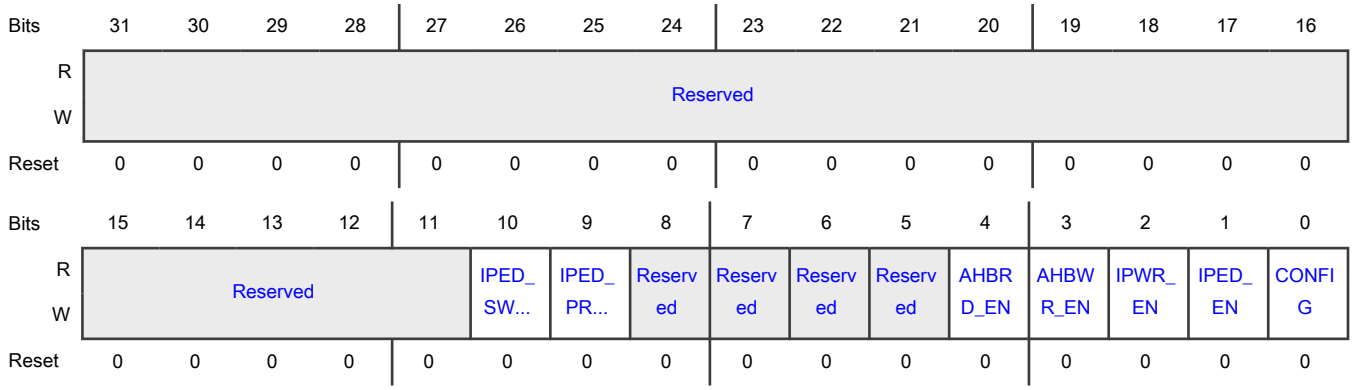
10.7.2.35 IPED Function Control (IPEDCTRL)

Offset

Register	Offset
IPEDCTRL	42Ch

Function
Selects IPED mode for encryption and decryption.

Diagram



Fields

Field	Function
31-11 —	Reserved
10 IPED_SWRESET	Abort Current Decryption or Encryption 0b - No function. 1b - Aborts current decryption or encryption and waits for the next start operation.
9 IPED_PROTECT	IPED Protection Can be used to limit access to IPED configuration registers. 0b - No restrictions 1b - Only privileged controllers can write IPED registers.
8 —	Reserved
7 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 —	Reserved
5 —	Reserved
4 AHBRD_EN	AHB Read IPED CTR Mode Decryption Enable 0b - Disable 1b - Enable
3 AHBWR_EN	AHB Write IPED CTR Mode Encryption Enable. 0b - Disable 1b - Enable
2 IPWR_EN	IP Write IPED CTR Mode Encryption Enable 0b - Disable 1b - Enable
1 IPED_EN	IPED Encryption and Decryption Enable 0b - Disable 1b - Enable
0 CONFIG	IPED Mode Select Configures whether encryption and decryption are fully pipelined. 0b - Fully pipelined 1b - Not fully pipelined

10.7.2.36 IPS Nonsecure Region a Start Address (IPSNSZSTART0 - IPSNSZSTART1)

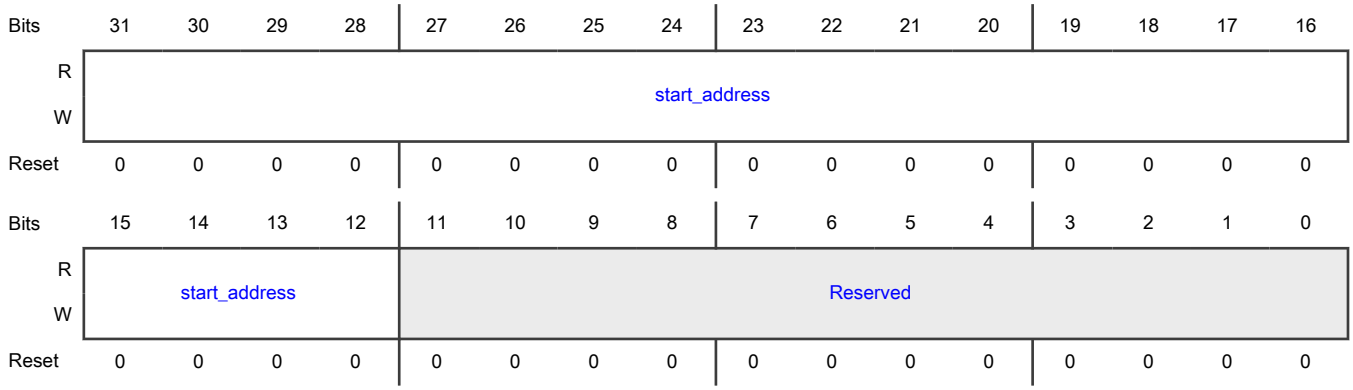
Offset

Register	Offset
IPSNSZSTART0	430h
IPSNSZSTART1	438h

Function

Indicates the start address of IPS nonsecure address region.

Diagram



Fields

Field	Function
31-12 start_address	Start Address of Nonsecure Region Indicates start address. Must be at least four-KB aligned. The address used here is the flash memory address that is compared with IPCR0[SFAR] to determine whether an IP command occurs in the region. Only a secure controller can write to this register. See Domain control.
11-0 —	Reserved

10.7.2.37 IPS Nonsecure Region a End Address (IPSNSZEND0 - IPSNSZEND1)

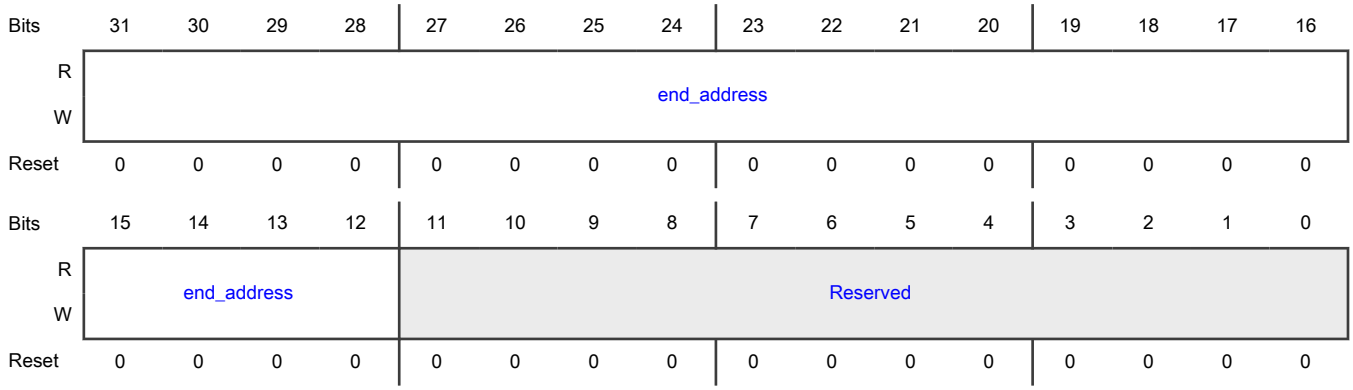
Offset

Register	Offset
IPSNSZEND0	434h
IPSNSZEND1	43Ch

Function

Indicates the end address of IPS nonsecure address region.

Diagram



Fields

Field	Function
31-12 end_address	End Address of Nonsecure Region Indicates end address. Must be at least four-KB aligned. The address used is the flash memory address compared with IPCR0[SFAR] to determine if an IP command hits in the region. Only a secure controller can write to this register. See Domain control.
11-0 —	Reserved

10.7.2.38 Receive Buffer Start Address of Region a (AHBBUFREGIONSTART0 - AHBBUFREGIONSTART3)

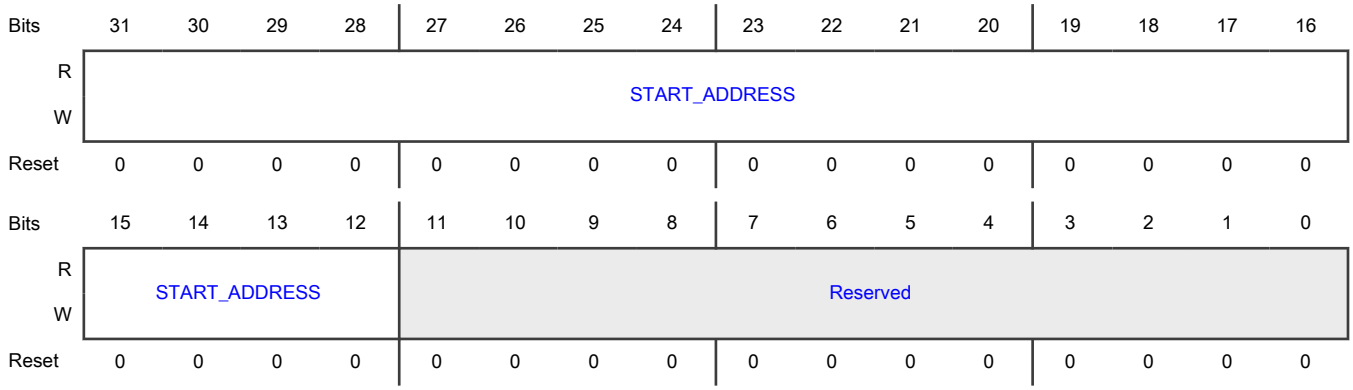
Offset

Register	Offset
AHBBUFREGIONSTART0	440h
AHBBUFREGIONSTART1	448h
AHBBUFREGIONSTART2	450h
AHBBUFREGIONSTART3	458h

Function

Indicates the start address of AHB receive buffer address region.

Diagram



Fields

Field	Function
31-12 START_ADDRESS	Start Address of Prefetch Sub-Buffer Region Indicates start address. The start address must be at least four-KB aligned. The address used here is the AHB address that is compared with the system bus address to determine if an AHB read hits in the region. See Prefetch buffer enhancement.
11-0 —	Reserved

10.7.2.39 Receive Buffer Region a End Address (AHBBUFREGIONEND0 - AHBBUFREGIONEND3)

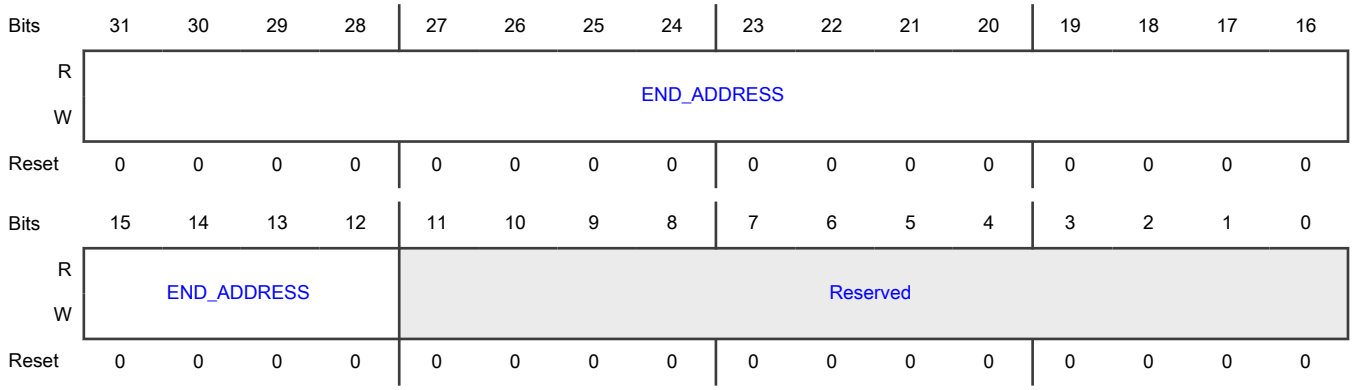
Offset

Register	Offset
AHBBUFREGIONEND0	444h
AHBBUFREGIONEND1	44Ch
AHBBUFREGIONEND2	454h
AHBBUFREGIONEND3	45Ch

Function

Indicates the end address of AHB receive buffer address region.

Diagram



Fields

Field	Function
31-12 END_ADDRESS	End Address of Prefetch Sub-Buffer Region Indicates end address. The end address must be at least four-KB aligned. The address used here is the AHB address that is compared with the system bus address to determine if an AHB read hits in the region. See Prefetch buffer enhancement.
11-0 —	Reserved

10.7.2.40 IPED context control 0 (IPEDCTXCTRL0)

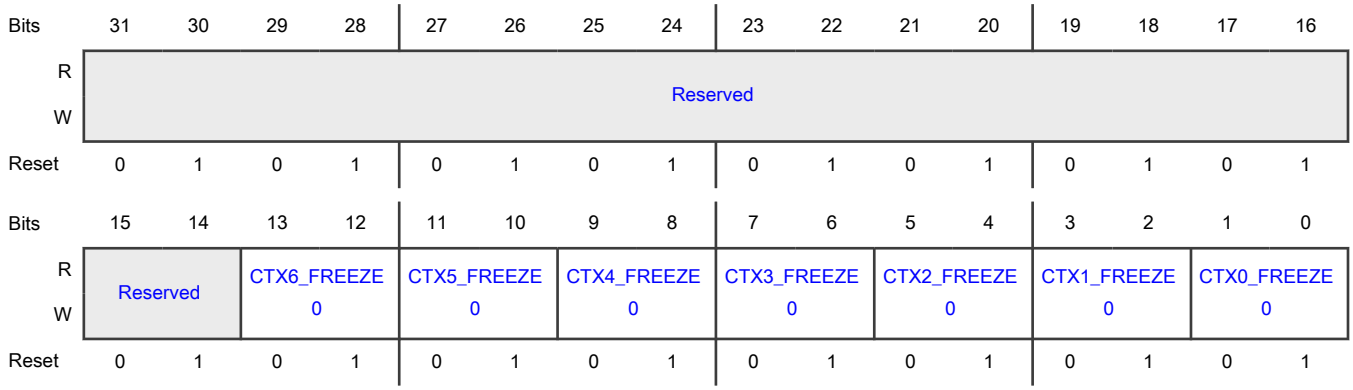
Offset

Register	Offset
IPEDCTXCTRL0	500h

Function

Freezes the IPEDCTXnSTART and IPEDCTXnEND registers. If bit1 and bit0 of the CTXn_FREEZE0 field are different, and IPEDCTXCTRL1[CTXn_FREEZE1] = 10b, CTXn_FREEZE0 is writable. See Inline PRINCE encryption and decryption (IPED) CTR.

Diagram



Fields

Field	Function
31-14 —	Reserved
13-12 CTX6_FREEZE0	Context Register Freeze for Region 6 Controls the read and write properties of this field and region 6 context registers (IPEDCTX6xxxx). This field freezes the IPEDCTX6START and IPEDCTX6END registers. If bit1 and bit0 of CTX6_FREEZE0 field are different, and IPEDCTXCTRL1[CTX6_FREEZE1] = 10b, CTX6_FREEZE0 is writable.
11-10 CTX5_FREEZE0	Context Register Freeze for Region 5 Controls the read and write properties of this field and region 5 context registers (IPEDCTX5xxxx). This field freezes the IPEDCTX5START and IPEDCTX5END registers. If bit1 and bit0 of CTX5_FREEZE0 field are different, and IPEDCTXCTRL1[CTX5_FREEZE1] = 10b, CTX5_FREEZE0 is writable.
9-8 CTX4_FREEZE0	Context Register Freeze for Region 4 Controls the read and write properties of this field and region 4 context registers (IPEDCTX4xxxx). This field freezes the IPEDCTX4START and IPEDCTX4END registers. If bit1 and bit0 of CTX4_FREEZE0 field are different, and IPEDCTXCTRL1[CTX4_FREEZE1] = 10b, CTX4_FREEZE0 is writable.
7-6 CTX3_FREEZE0	Context Register Freeze for Region 3 Controls the read and write properties of this field and region 3 context registers (IPEDCTX3xxxx). This field freezes the IPEDCTX3START and IPEDCTX3END registers. If bit1 and bit0 of CTX3_FREEZE0 field are different, and IPEDCTXCTRL1[CTX3_FREEZE1] = 10b, CTX3_FREEZE0 is writable.
5-4 CTX2_FREEZE0	Context Register Freeze for Region 2 Controls the read and write properties of this field and region 2 context registers (IPEDCTX2xxxx). This field freezes the IPEDCTX2START and IPEDCTX2END registers. If bit1 and bit0 of CTX2_FREEZE0 field are different, and IPEDCTXCTRL1[CTX2_FREEZE1] = 10b, CTX2_FREEZE0 is writable.
3-2 CTX1_FREEZE0	Context Register Freeze for Region 1 Controls the read and write properties of this field and region 1 context registers (IPEDCTX1xxxx). This field freezes the IPEDCTX1START and IPEDCTX1END registers. If bit1 and bit0 of CTX1_FREEZE0 field are different, and IPEDCTXCTRL1[CTX1_FREEZE1] = 10b, CTX1_FREEZE0 is writable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 CTX0_FREEZE0	Context Register Freeze for Region 0 Controls the read and write properties of this field and region 0 context registers (IPEDCTX0xxxx). This field freezes the IPEDCTX0START and IPEDCTX0END registers. If bit1 and bit0 of CTX0_FREEZE0 field are different, and IPEDCTXCTRL1[CTX0_FREEZE1] = 10b, CTX0_FREEZE0 is writable.

10.7.2.41 IPED context control 1 (IPEDCTXCTRL1)

Offset

Register	Offset
IPEDCTXCTRL1	504h

Function

Freezes the IPEDCTXnSTART and IPEDCTXnEND registers. If bit1 and bit0 of the CTXn_FREEZE0 field are different, and IPEDCTXCTRL1[CTXn_FREEZE1] = 10b, CTXn_FREEZE1 is writable. See Inline PRINCE encryption and decryption (IPED) CTR.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		CTX6_FREEZE1		CTX5_FREEZE1		CTX4_FREEZE1		CTX3_FREEZE1		CTX2_FREEZE1		CTX1_FREEZE1		CTX0_FREEZE1	
Reset	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Fields

Field	Function
31-14 —	Reserved
13-12 CTX6_FREEZE1	Context Register Freeze for Region 6 Controls the read and write properties of this field and region 6 context registers (IPEDCTX6xxxx). This field freezes the IPEDCTX6START and IPEDCTX6END registers. If bit1 and bit0 of CTX6_FREEZE0 field are different, and IPEDCTXCTRL1[CTX6_FREEZE1] = 10b, CTX6_FREEZE1 is writable.
11-10	Context Register Freeze for Region 5

Table continues on the next page...

Table continued from the previous page...

Field	Function
CTX5_FREEZE1	Controls the read and write properties of this field and region 5 context registers (IPEDCTX5xxxx). This field freezes the IPEDCTX5START and IPEDCTX5END registers. If bit1 and bit0 of CTX5_FREEZE0 field are different, and IPEDCTXCTRL1[CTX5_FREEZE1] = 10b, CTX5_FREEZE1 is writable.
9-8 CTX4_FREEZE1	Context Register Freeze for Region 4 Controls the read and write properties of this field and region 4 context registers (IPEDCTX4xxxx). This field freezes the IPEDCTX4START and IPEDCTX4END registers. If bit1 and bit0 of CTX4_FREEZE0 field are different, and IPEDCTXCTRL1[CTX4_FREEZE1] = 10b, CTX4_FREEZE1 is writable.
7-6 CTX3_FREEZE1	Context Register Freeze for Region 3 Controls the read and write properties of this field and region 3 context registers (IPEDCTX3xxxx). This field freezes the IPEDCTX3START and IPEDCTX3END registers. If bit1 and bit0 of CTX3_FREEZE0 field are different, and IPEDCTXCTRL1[CTX3_FREEZE1] = 10b, CTX3_FREEZE1 is writable.
5-4 CTX2_FREEZE1	Context Register Freeze for Region 2 Controls the read and write properties of this field and region 2 context registers (IPEDCTX2xxxx). This field freezes the IPEDCTX2START and IPEDCTX2END registers. If bit1 and bit0 of CTX2_FREEZE0 field are different, and IPEDCTXCTRL1[CTX2_FREEZE1] = 10b, CTX2_FREEZE1 is writable.
3-2 CTX1_FREEZE1	Context Register Freeze for Region 1 Controls the read and write properties of this field and region 1 context registers (IPEDCTX1xxxx). This field freezes the IPEDCTX1START and IPEDCTX1END registers. If bit1 and bit0 of CTX1_FREEZE0 field are different, and IPEDCTXCTRL1[CTX1_FREEZE1] = 10b, CTX1_FREEZE1 is writable.
1-0 CTX0_FREEZE1	Context Register Freeze for Region 0 Controls the read and write properties of this field and region 0 context registers (IPEDCTX0xxxx). This field freezes the IPEDCTX0START and IPEDCTX0END registers. If bit1 and bit0 of CTX0_FREEZE0 field are different, and IPEDCTXCTRL1[CTX0_FREEZE1] = 10b, CTX0_FREEZE1 is writable.

10.7.2.42 IPED Context0 IV0 (IPEDCTX0IV0)

Offset

Register	Offset
IPEDCTX0IV0	520h

Function
Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX0_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX0_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX0_IV0	Lowest 32 bits of IV for region 0.

10.7.2.43 IPED Context0 IV1 (IPEDCTX0IV1)

Offset

Register	Offset
IPEDCTX0IV1	524h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX0_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX0_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX0_IV1	Highest 32 bits of IV for region 0.

10.7.2.44 Start Address of Region (IPEDCTX0START - IPEDCTX6START)

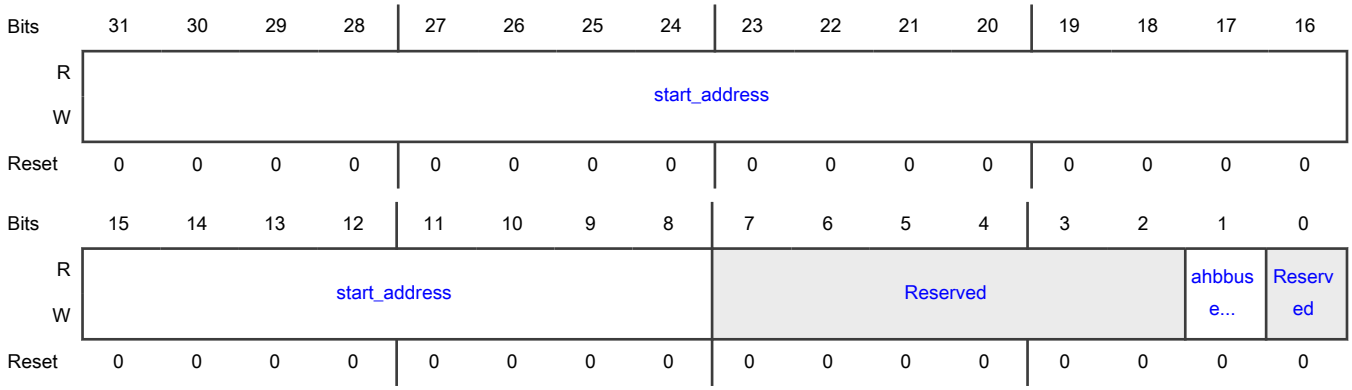
Offset

Register	Offset
IPEDCTX0START	528h
IPEDCTX1START	548h
IPEDCTX2START	568h
IPEDCTX3START	588h
IPEDCTX4START	5A8h
IPEDCTX5START	5C8h
IPEDCTX6START	5E8h

Function

Indicates start address of region. If the CTXn_FREEZE0 fields of the IPEDCTXCTRL0 register are 10b and CTXn_FREEZE1 fields of IPEDCTXCTRL1 register are 10b, then IPEDCTXnSTART is writable.

Diagram



Fields

Field	Function
31-8 start_address	Start Address

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates start address of IPED region. The start address must be at least 256-byte aligned. The address used here is the AHB address that is compared with the system bus address.
7-2 —	Reserved
1 ahbbuserror_dis	AHB Bus Error Disable Used to configure whether IPED errors generate bus errors on an AHB access. 0b - AHB bus errors enabled 1b - AHB bus errors disabled
0 —	Reserved

10.7.2.45 End Address of Region (IPEDCTX0END - IPEDCTX6END)

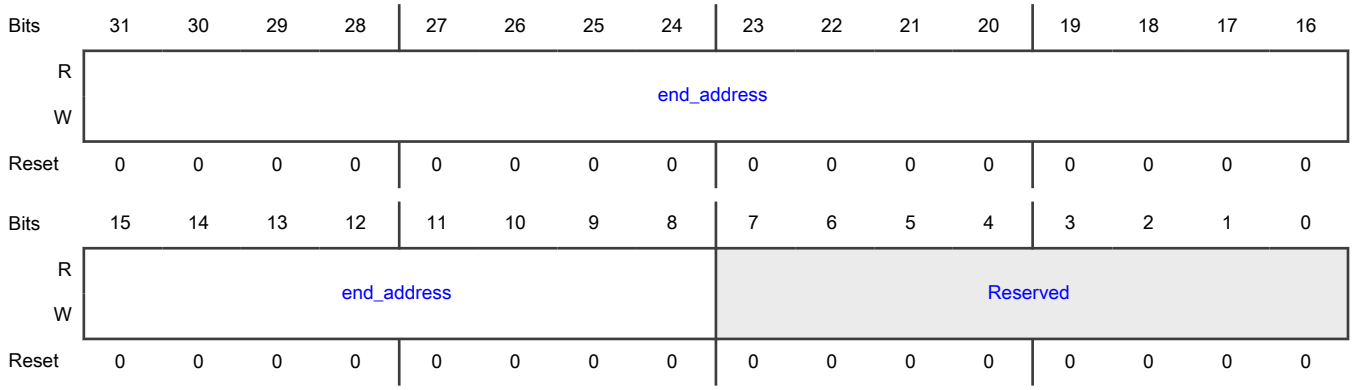
Offset

Register	Offset
IPEDCTX0END	52Ch
IPEDCTX1END	54Ch
IPEDCTX2END	56Ch
IPEDCTX3END	58Ch
IPEDCTX4END	5ACh
IPEDCTX5END	5CCh
IPEDCTX6END	5ECh

Function

Indicates end address of region. If IPEDCTXCTRL0[CTXn_FREEZE0] = 10b and IPEDCTXCTRL1[CTXn_FREEZE1] = 10b, IPEDCTXnEND is writable.

Diagram



Fields

Field	Function
31-8 end_address	End Address of IPED Region Contains the end address of the IPED region. Must be at least 256-byte aligned. This address is the AHB address that is compared with the system bus address.
7-0 —	Reserved

10.7.2.46 IPED Context1 IV0 (IPEDCTX1IV0)

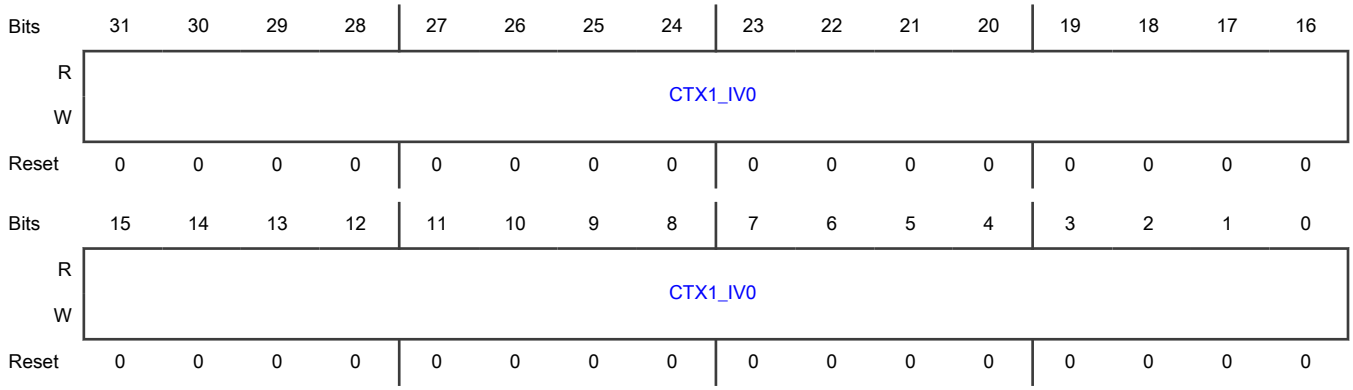
Offset

Register	Offset
IPEDCTX1IV0	540h

Function

Sets the IPED context IV for encryption and decryption.

Diagram



Fields

Field	Function
31-0 CTX1_IV0	Lowest 32 bits of IV for region 1.

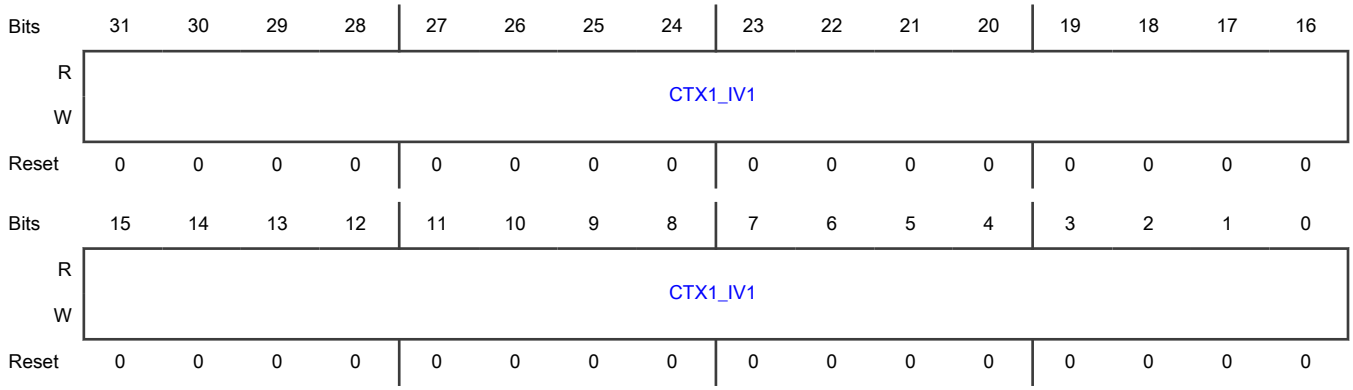
10.7.2.47 IPED Context1 IV1 (IPEDCTX1IV1)

Offset

Register	Offset
IPEDCTX1IV1	544h

Function
Sets the IPED context IV for encryption and decryption.

Diagram



Fields

Field	Function
31-0 CTX1_IV1	Highest 32 bits of IV for region 1.

10.7.2.48 IPED Context2 IV0 (IPEDCTX2IV0)

Offset

Register	Offset
IPEDCTX2IV0	560h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX2_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX2_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX2_IV0	Lowest 32 bits of IV for region 2.

10.7.2.49 IPED Context2 IV1 (IPEDCTX2IV1)

Offset

Register	Offset
IPEDCTX2IV1	564h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX2_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX2_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX2_IV1	Highest 32 bits of IV for region 2.

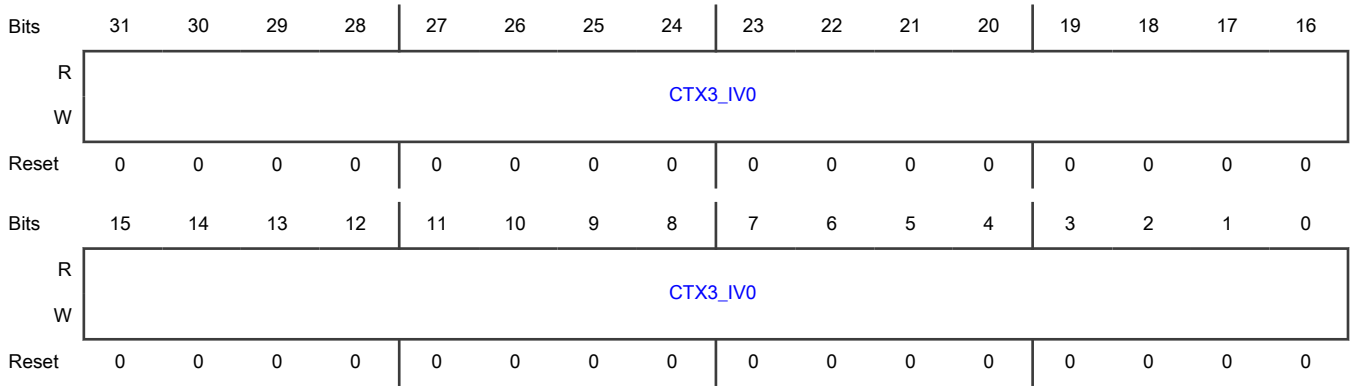
10.7.2.50 IPED Context3 IV0 (IPEDCTX3IV0)

Offset

Register	Offset
IPEDCTX3IV0	580h

Function
Sets the IPED context IV for encryption and decryption.

Diagram



Fields

Field	Function
31-0 CTX3_IV0	Lowest 32 bits of IV for region 3.

10.7.2.51 IPED Context3 IV1 (IPEDCTX3IV1)

Offset

Register	Offset
IPEDCTX3IV1	584h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX3_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX3_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX3_IV1	Highest 32 bits of IV for region 3.

10.7.2.52 IPED Context4 IV0 (IPEDCTX4IV0)

Offset

Register	Offset
IPEDCTX4IV0	5A0h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX4_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX4_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX4_IV0	Lowest 32 bits of IV for region 4.

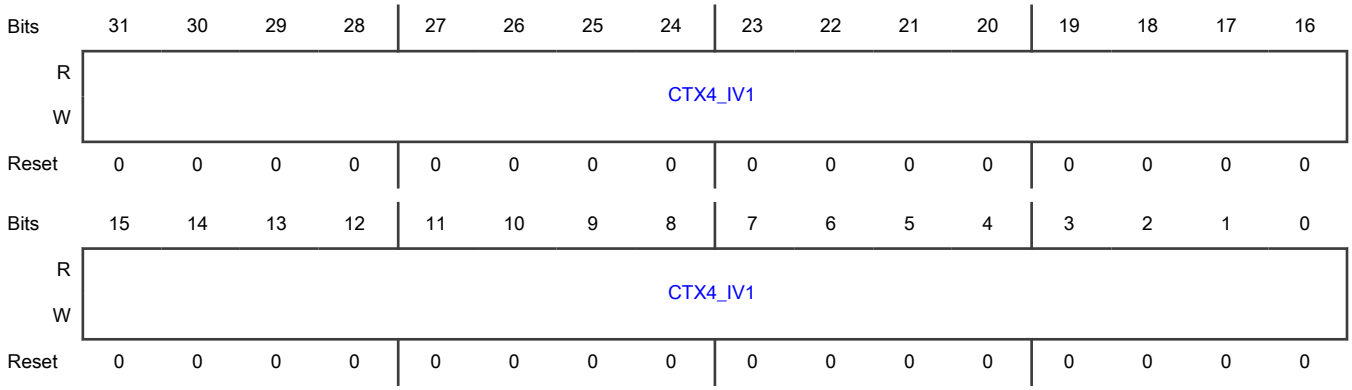
10.7.2.53 IPED Context4 IV1 (IPEDCTX4IV1)

Offset

Register	Offset
IPEDCTX4IV1	5A4h

Function
Sets the IPED context IV for encryption and decryption.

Diagram



Fields

Field	Function
31-0 CTX4_IV1	Highest 32 bits of IV for region 4.

10.7.2.54 IPED Context5 IV0 (IPEDCTX5IV0)

Offset

Register	Offset
IPEDCTX5IV0	5C0h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX5_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX5_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX5_IV0	Lowest 32 bits of IV for region 5.

10.7.2.55 IPED Context5 IV1 (IPEDCTX5IV1)

Offset

Register	Offset
IPEDCTX5IV1	5C4h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX5_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTX5_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX5_IV1	Highest 32 bits of IV for region 5.

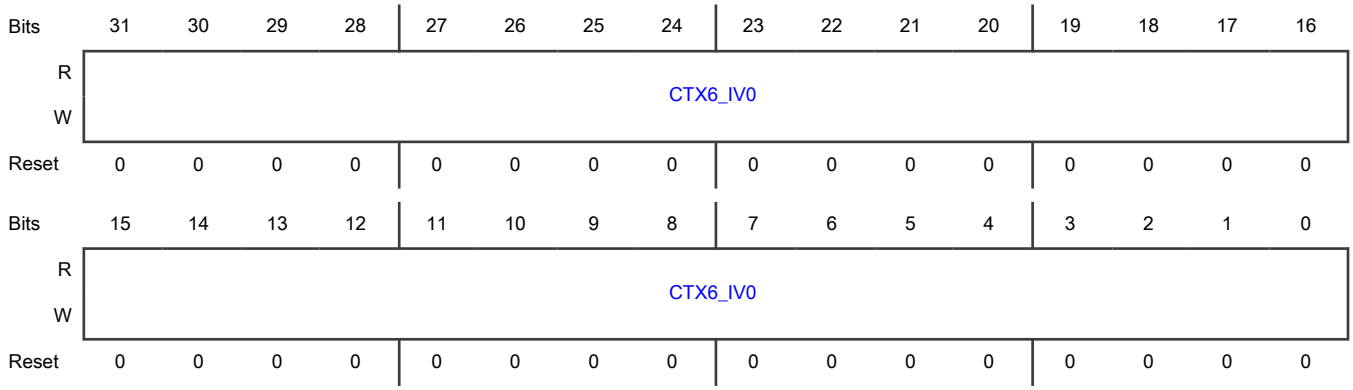
10.7.2.56 IPED Context6 IV0 (IPEDCTX6IV0)

Offset

Register	Offset
IPEDCTX6IV0	5E0h

Function
Sets the IPED context IV for encryption and decryption.

Diagram



Fields

Field	Function
31-0 CTX6_IV0	Lowest 32 bits of IV for region 6.

10.7.2.57 IPED Context6 IV1 (IPEDCTX6IV1)

Offset

Register	Offset
IPEDCTX6IV1	5E4h

Function

Sets the IPED context IV for encryption and decryption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CTX6_IV1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CTX6_IV1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CTX6_IV1	Highest 32 bits of IV for region 6.

10.8 AHB memory map definition

This section describes FlexSPI module AHB memory map in detail.

10.8.1 AHB memory map for serial flash memory and RAM access

AHB read and write access for serial flash and RAM memory are mapped to a specific address range. See the system memory map for specific address ranges supported.

AHB bus features supported for serial flash and RAM memory reading:

- Cacheable and non-cacheable access
- Prefetch enable and disable
- Burst size: 8, 16, 32, 64 bits
- All burst types: types: SINGLE, INCR, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16

AHB bus features for Serial RAM writing:

- Bufferable and Non-Bufferable access
- Burst size: 8, 16, 32, 64 bits
- All burst types: SINGLE, INCR, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16

See Flash memory access via AHB command for details about AHB access to serial flash memory and RAM.

Chapter 11
Performance Monitor (CMX_PERFMON)

11.1 Chip-specific CMX_PERFMON information

Table 116. Reference links to related information

Topic	Related module	Reference
Full description	CMX_PERFMON	CMX_PERFMON
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

11.1.1 Module instances

This device has two instances of the CMX_PERFMON module. Two performance monitors are used in this device; one is for monitoring LPCAC that maps to CMX_PERFMON0, and another one is for FlexSPI that maps to CMX_PERFMON1.

11.1.2 Perfmon Counters

The following table describes the countable events for the LPCAC and FlexSPI performance monitors.

Table 117. Perfmon Counters

Event Encoding	Mnemonic	CMX_PERFMON Note: Same event encodings are used for CMX_PERFMON0 and CMX_PERFMON1 except cac_wb_push.	Notes
AHB nonTCM ("cache") Transfers			
0b100_0000	cac_if	YES	Cached instruction fetches
0b100_0100	cac_if_stalls	YES	Stalled cached instruction fetches
0b100_0001	cac_opr	YES	Cached operand reads
0b100_0101	cac_opr_stalls	YES	Stalled cached operand reads
0b100_0010	cac_opw	YES	Cached operand writes
0b100_0110	cac_opw_stalls	YES	Stalled cached operand writes
0b100_0011	cac_opx	YES	Cached operand reads plus writes
0b100_0111	cac_opx_stalls	YES	Stalled cached operand reads plus writes
0b100_1000	cac_wt_opw	YES	Cache writethrough operand writes

Table continues on the next page...

Table 117. Perfmon Counters (continued)

Event Encoding	Mnemonic	CMX_PERFMON Note: Same event encodings are used for CMX_PERFMON0 and CMX_PERFMON1 except cac_wb_push.	Notes
0b100_1100	cac_wt_opw_stalls	YES	Stalled writethrough operand writes
0b101_0000	cac_if_miss	YES	Instruction fetch miss (for CMX_PERFMON1 - number of times FlexSPI signalled miss)
0b101_0001	cac_opr_miss	YES	Operand read miss
0b101_0010	cac_opw_miss	YES	Operand write miss
0b101_0011	cac_opx_miss	YES	Operand read plus write miss
0b101_1000	cac_wb_push	YES for CMX_PERFMON1, NO for CMX_PERFMON0	Cache writeback line pushes

11.2 Overview

CMX_PERFMON contains counters that you can configure to count events used to calculate the performance of a CPU, cache, or memory. You can configure CMX_PERFMON to select the events to be counted for each counter.

Block diagram provides an overview of CMX_PERFMON, which includes three event counters.

11.2.1 Block diagram

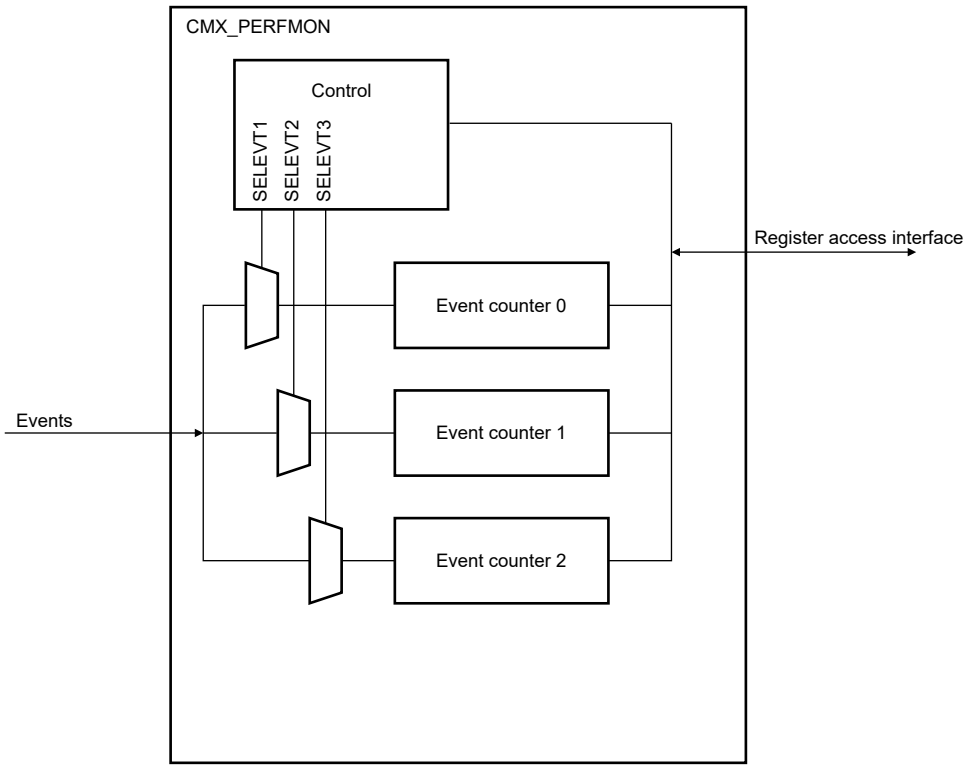


Figure 53. Block diagram

11.2.2 Features

- Three 40-bit event counters
- Programmable event select
- Countable events that include cache, branch fetches, and stalls
- Memory mapped performance monitor that counts instructions, cache, branch, IPS, and TCM stats
- Count filter based on privilege level

11.3 Functional description

11.3.1 Operations

See the chip-specific CMX_PERFMON information for details about the countable events for the performance counters per CMX_PERFMON instance on this chip.

CMX_PERFMON has 3 programmable counters to count CPU events. The event to be counted is configured using the PMCR0 register.

Each counter is a 40-bit register that tracks the number of occurrences of the selected event.

The counters can be programmed to take one of the 3 states:

- At idle state, the counters are in idle and do not perform any operation.

- At local start state, the counters are actively listening to CPU and cache and start incrementing when the selected events are observed.
- At local stop state, the counters stop listening to all components. The 3 event counter can be independently cleared by writing to PMCR0[RECTR*n*], where *n* denotes the counter instance.

11.3.2 Operating modes

The following table shows:

- The operating modes that you can specify for CMX_PERFMON.
- The PMCR0[CMODE] value that you must specify to count events in each mode.

Table 118. Operating modes

Mode	Description	CMODE value (binary)
Privileged	If supported, only counts events in Privilege mode.	11
User	If supported, only counts events in User mode.	10
Both Privileged and User	Counts events in both Privilege and User mode.	00

Use PMCR0[SSC] to start and stop the counters.

11.3.3 Clocking

This module has no clocking considerations.

11.3.4 Interrupts

This module has no interrupts.

11.4 External signals

CMX_PERFMON has no external signals.

11.5 Initialization

To initialize CMX_PERFMON:

1. Program PMCR0[SELEVT*n*] to select the specific CPU event to be counted by this counter, where *n* denotes the counter instance.
2. Set PMCR0[CMODE] to filter each counter for the selected event while the CPU is in privilege, user, or either mode.
3. Set PMCR0[SSC] to program the counter to be in one of idle, local start, or local stop states.
4. Set PMCR0[RECTR*n*] to clear the counters independently where *n* denotes the counter instance.

11.6 Memory map and register definition

11.6.1 CMX_PERFMON register descriptions

11.6.1.1 CMX_PERFMON memory map

CMX_PERFMON0 base address: 400C_1000h

CMX_PERFMON1 base address: 400C_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Performance Monitor Control (PMCR)	32	RW	0000_0000h
18h	Performance Monitor Event Counter (PMECTR1_HI)	8	R	00h
1Ch	Performance Monitor Event Counter (PMECTR1_LO)	32	R	0000_0000h
20h	Performance Monitor Event Counter (PMECTR2_HI)	8	R	00h
24h	Performance Monitor Event Counter (PMECTR2_LO)	32	R	0000_0000h
28h	Performance Monitor Event Counter (PMECTR3_HI)	8	R	00h
2Ch	Performance Monitor Event Counter (PMECTR3_LO)	32	R	0000_0000h

11.6.1.2 Performance Monitor Control (PMCR)

Offset

Register	Offset
PMCR	0h

Function

Specifies:

- The events that the PMECTR*n*HI and PMECTR*n*LO count.
- The count mode.
- The start and stop control.
- The enables for the counters.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SELEVT3								SELEVT2						SELEVT1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SELEVT1					RECTR3	RECTR2	RECTR1	Reserved	Reserved	CMODE		SSC			MENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 SELEVT3	Select Event 3 Selects the event to be counted in PMECTR3. See the chip-specific CMX_PERFMON information to select which event PMECTR3 counts.
24-18 SELEVT2	Select Event 2 Selects the event to be counted in PMECTR2. See the chip-specific CMX_PERFMON information to select which event PMECTR2 counts.
17-11 SELEVT1	Select Event 1 Selects the event to be counted in PMECTR1. See the chip-specific CMX_PERFMON information to select which event PMECTR1 counts.
10 RECTR3	Reset Event Counter 3 Specifies whether the counter runs normally or the counter value resets at the end of the cycle. Write 1 to this field to clear this counter. This field does not return to 0 automatically, so if you want to restart the counter after clearing it, write 0 to this field. 0b - Run normally 1b - Reset
9 RECTR2	Reset Event Counter 2 Specifies whether the counter runs normally or the counter value resets at the end of the cycle. Write 1 to this field to clear this counter. This field does not return to 0 automatically, so if you want to restart the counter after clearing it, write 0 to this field. 0b - Run normally 1b - Reset
8 RECTR1	Reset Event Counter 1 Specifies whether the counter runs normally or the counter value resets at the end of the cycle. Write 1 to this field to clear this counter. This field does not return to 0 automatically, so if you want to restart the counter after clearing it, write 0 to this field. 0b - Run normally 1b - Reset
7 —	Reserved
6 —	Reserved
5-4	Count Mode

Table continues on the next page...

Table continued from the previous page...

Field	Function
CMODE	Determines whether events are counted for Privileged mode, User mode, or both Privileged and User modes. This setting affects the operation of all event counters. 00b - Counted in both User and Privileged modes 01b - Reserved 10b - Reserved 11b - Reserved
3-1 SSC	Start and Stop Control Provides a three-phase mechanism to start and stop the counters. It includes a prioritized scheme with the following relative priorities: local start > local stop The start or stop affects all performance monitor event counters concurrently, so that the counters are coherent. 000b - Idle or no-op 001b - Local stop 010b,011b - Local start 100b - Reserved 101b - Reserved 110b,111b - Reserved
0 MENB	Module Is Enabled Indicates whether CMX_PERFMON is enabled. Enabling CMX_PERFMON does not start the counters. To start the counters, write a nonzero value to PMCR0[SSC]. 0b - Disabled 1b - Enabled

11.6.1.3 Performance Monitor Event Counter (PMECTR1_HI - PMECTR3_HI)

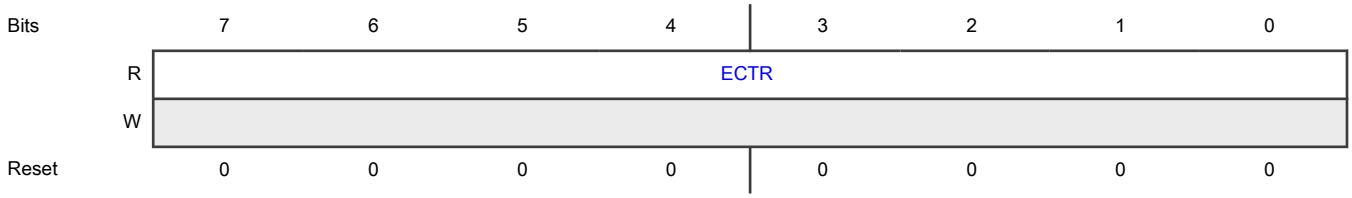
Offset

Register	Offset
PMECTR1_HI	18h
PMECTR2_HI	20h
PMECTR3_HI	28h

Function

Provides part of a 40-bit event counter that you configure with PMCR0[SELEVT*n*].

Diagram



Fields

Field	Function
7-0 ECTR	Event Counter Specifies the upper 8 bits of the event[*n*] counter.

11.6.1.4 Performance Monitor Event Counter (PMECTR1_LO - PMECTR3_LO)

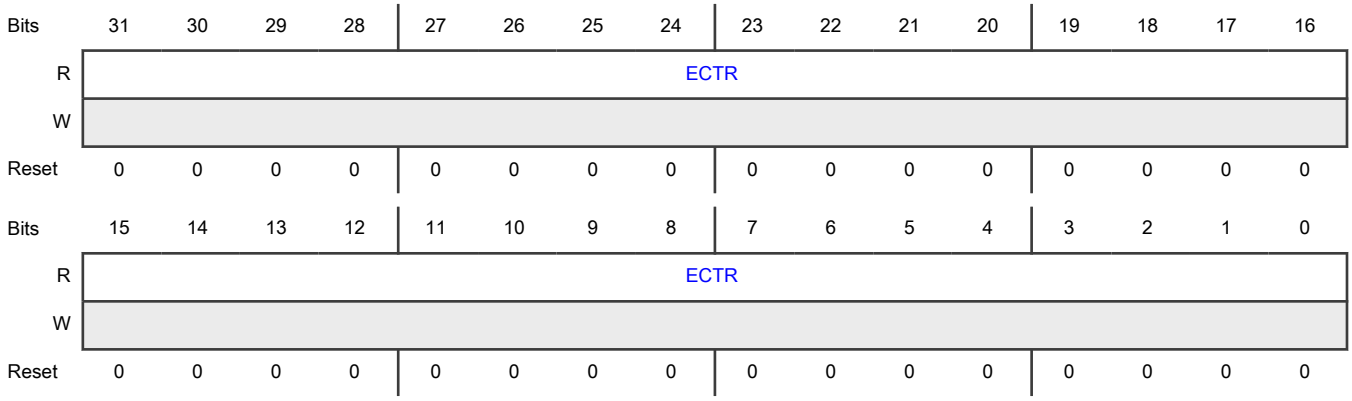
Offset

Register	Offset
PMECTR1_LO	1Ch
PMECTR2_LO	24h
PMECTR3_LO	2Ch

Function

Provides part of a 40-bit event counter that you configure with PMCR0[SELEVT*n*].

Diagram



Fields

Field	Function
31-0	Event Counter

Table continues on the next page...

Field	Function
ECTR	Specifies the lower 32 bits of the event[*n*] counter. The 40-bit counter value increments each time the event selected in [SELEVT*n*] occurs.

Chapter 12
Signal Multiplexing

12.1 Introduction

Pins have several functions available via signal multiplexing to optimize functionality in small packages. This chapter illustrates which device's signals are multiplexed on which external pin.

See the pinout table (also in the attached spreadsheet) to know how the pins available on this device are configured. The attached spreadsheet also provides the available device packages and pinout diagrams.

12.1.1 Signal multiplexing constraints

- A given module signal must be assigned to a maximum of one package pin. Do not program the same function to more than one pin.
- To ensure the best signal timing for a given peripheral's interface, choose the pins in closest proximity to each other.

12.2 Pinout

See the following table for pinout details, also in the attached MCXNP184M150F70_Pinout.xlsx

12.2.1 Pinmux view

Table 119. pinmux

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
P1_8	A1	172	1	1	**ALT0** - P1_8 **ALT1** - TRACE_DATA0 **ALT2** - FC4_P0 **ALT3** - FC5_P4 **ALT4** - CT_INP8 **ALT5** - SCT0_OUT2 **ALT6** - FLEXIO0_D16 **ALT7** - SMARTDMA_PIO4 **ALT8** - PLU_OUT0 **ALT9** - ENET0_TXD2 **ALT10** - I3C1_SDA	**IO Supply** - VDD **Pad type** - MED+I2C+I3C **Default** - DIS	**ISP** - UART_RXD **ANALOG** - TSI0_CH17/ADC1_A8 **VDD SYS** - WUU0_IN10/ LPTMR1_ALT3
P1_9	B1	1	2	2	**ALT0** - P1_9 **ALT1** - TRACE_DATA1 **ALT2** - FC4_P1 **ALT3** - FC5_P5 **ALT4** - CT_INP9 **ALT5** - SCT0_OUT3	**IO Supply** - VDD **Pad type** - MED+I2C **Default** - DIS	**ISP** - UART_TXD **ANALOG** - TSI0_CH18/ADC1_A9

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT6 - FLEXIO0_D17 **ALT7** - SMARTDMA_PIO5 **ALT8** - PLU_OUT1 **ALT9** - ENET0_TXD3 **ALT10** - I3C1_SCL		
P1_10	C3	2	3	3	**ALT0** - P1_10 **ALT1** - TRACE_DATA2 **ALT2** - FC4_P2 **ALT3** - FC5_P6 **ALT4** - CT2_MAT0 **ALT5** - SCT0_IN2 **ALT6** - FLEXIO0_D18 **ALT7** - SMARTDMA_PIO6 **ALT8** - PLU_IN0 **ALT9** - ENET0_TXER **ALT11** - CAN0_TXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ISP** - CAN_TXD **ANALOG** - TSI0_CH19/ADC1_A10
P1_11	D3	3	4	4	**ALT0** - P1_11 **ALT1** - TRACE_DATA3 **ALT2** - FC4_P3 **ALT4** - CT2_MAT1 **ALT5** - SCT0_IN3 **ALT6** - FLEXIO0_D19 **ALT7** - SMARTDMA_PIO7 **ALT8** - PLU_IN1 **ALT9** - ENET0_RX_CLK **ALT10** - I3C1_PUR **ALT11** - CAN0_RXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ISP** - CAN_RXD **ANALOG** - TSI0_CH20/ADC1_A11 **VDD SYS** - WUU0_IN11
P1_12	D2	4	5	5	**ALT0** - P1_12 **ALT1** - TRACE_CLK **ALT2** - FC4_P4 **ALT3** - FC3_P0 **ALT4** - CT2_MAT2 **ALT5** - SCT0_OUT4 **ALT6** - FLEXIO0_D20	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH21/ADC1_A12 **VDD SYS** - WUU0_IN12

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT7 - SMARTDMA_PIO8 **ALT8** - PLU_OUT2 **ALT9** - ENET0_RXER **ALT11** - CAN1_RXD		
P1_13	D1	5	6	6	**ALT0** - P1_13 **ALT1** - TRIG_IN3 **ALT2** - FC4_P5 **ALT3** - FC3_P1 **ALT4** - CT2_MAT3 **ALT5** - SCT0_OUT5 **ALT6** - FLEXIO0_D21 **ALT7** - SMARTDMA_PIO9 **ALT8** - PLU_OUT3 **ALT9** - ENET0_RXDV **ALT11** - CAN1_TXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH22/ADC1_A13
P1_14	D4	6	7	7	**ALT0** - P1_14 **ALT2** - FC4_P6 **ALT3** - FC3_P2 **ALT4** - CT_INP10 **ALT5** - SCT0_IN4 **ALT6** - FLEXIO0_D22 **ALT7** - SMARTDMA_PIO10 **ALT8** - PLU_IN2 **ALT9** - ENET0_RXD0	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH23/ADC1_A14
P1_15	E4	7	8	8	**ALT0** - P1_15 **ALT3** - FC3_P3 **ALT4** - CT_INP11 **ALT5** - SCT0_IN5 **ALT6** - FLEXIO0_D23 **ALT7** - SMARTDMA_PIO11 **ALT8** - PLU_IN3 **ALT9** - ENET0_RXD1 **ALT10** - I3C1_PUR	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH24/ADC1_A15 **VDD SYS** - WUU0_IN13

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
VSS	P14	--	--	--		**IO Supply** - VDD **Pad type** - VSSIO	
P1_16	F6	8	--	--	**ALT0** - P1_16 **ALT2** - FC5_P0 **ALT3** - FC3_P4 **ALT4** - CT_INP12 **ALT5** - SCT0_OUT6 **ALT6** - FLEXIO0_D24 **ALT7** - SMARTDMA_PIO12 **ALT8** - PLU_OUT4 **ALT9** - ENET0_RXD2 **ALT10** - I3C1_SDA	**IO Supply** - VDD **Pad type** - MED+I2C+I3C **Default** - DIS	**ANALOG** - ADC1_A16 **VDD SYS** - WUU0_IN14
P1_17	F4	9	--	--	**ALT0** - P1_17 **ALT2** - FC5_P1 **ALT3** - FC3_P5 **ALT4** - CT_INP13 **ALT5** - SCT0_OUT7 **ALT6** - FLEXIO0_D25 **ALT7** - SMARTDMA_PIO13 **ALT8** - PLU_OUT5 **ALT9** - ENET0_RXD3 **ALT10** - I3C1_SCL	**IO Supply** - VDD **Pad type** - MED+I2C **Default** - DIS	**ANALOG** - ADC1_A17
P1_18	G4	10	--	--	**ALT0** - P1_18 **ALT1** - FREQME_CLK_IN0 **ALT2** - FC5_P2 **ALT3** - FC3_P6 **ALT4** - CT3_MAT0 **ALT5** - SCT0_IN6 **ALT6** - FLEXIO0_D26 **ALT7** - SMARTDMA_PIO14 **ALT8** - PLU_IN4 **ALT9** - ENET0_COL **ALT11** - CAN0_TXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC1_A18

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
P1_19	G5	11	--	--	**ALT0** - P1_19 **ALT1** - FREQME_CLK_IN1 **ALT2** - FC5_P3 **ALT4** - CT3_MAT1 **ALT5** - SCT0_IN7 **ALT6** - FLEXIO0_D27 **ALT7** - SMARTDMA_PIO15 **ALT8** - PLU_IN5 **ALT9** - ENET0_CRS **ALT11** - CAN0_RXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC1_A19 **VDD SYS** - WUU0_IN15
P1_20	K5	--	--	--	**ALT0** - P1_20 **ALT1** - TRIG_IN2 **ALT2** - FC5_P4 **ALT3** - FC4_P0 **ALT4** - CT3_MAT2 **ALT5** - SCT0_OUT8 **ALT6** - FLEXIO0_D28 **ALT7** - SMARTDMA_PIO16 **ALT8** - PLU_OUT6 **ALT9** - ENET0_MDC **ALT11** - CAN1_TXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC1_A20/CMP1_IN3
P1_21	L5	--	--	--	**ALT0** - P1_21 **ALT1** - TRIG_OUT2 **ALT2** - FC5_P5 **ALT3** - FC4_P1 **ALT4** - CT3_MAT3 **ALT5** - SCT0_OUT9 **ALT6** - FLEXIO0_D29 **ALT7** - SMARTDMA_PIO17 **ALT8** - PLU_OUT7 **ALT9** - ENET0_MDIO **ALT10** - SAI1_MCLK **ALT11** - CAN1_RXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC1_A21/CMP2_IN3

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
P1_22	L4	--	--	--	**ALT0** - P1_22 **ALT1** - TRIG_IN3 **ALT2** - FC5_P6 **ALT3** - FC4_P2 **ALT4** - CT_INP14 **ALT5** - SCT0_OUT4 **ALT6** - FLEXIO0_D30 **ALT7** - SMARTDMA_PIO18	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC1_A22
P1_23	M4	--	--	--	**ALT0** - P1_23 **ALT3** - FC4_P3 **ALT4** - CT_INP15 **ALT5** - SCT0_OUT5 **ALT6** - FLEXIO0_D31 **ALT7** - SMARTDMA_PIO19	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC1_A23
RESET_B	F3	12	9	9		**IO Supply** - VDD **Pad type** - RST **Default** - RESET_B	
P1_30	F1	13	10	10	**ALT0** - P1_30 **ALT1** - TRIG_OUT3 **ALT4** - CT_INP16 **ALT5** - SCT0_OUT8 **ALT10** - SAI0_MCLK	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - XTAL48M
P1_31	F2	14	11	11	**ALT0** - P1_31 **ALT1** - TRIG_IN4 **ALT4** - CT_INP17 **ALT5** - SCT0_OUT9	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - EXTAL48M
VSS	D6	15	0	0		**IO Supply** - VDD **Pad type** - VSSIO	
VDD_CORE	K10	16	12	12		**IO Supply** - VDD **Pad type** - VDDINT	
VDD_LDO_CORE	K6	17	13	13		**IO Supply** - VDD **Pad type** - VDDINT_3V	**ANALOG** - VDD_LDO_CORE
VDD	H8	18	13	13		**IO Supply** - VDD	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
						Pad type - VDDIO	
VDD_P2	K8	19	13	13		**IO Supply** - VDD_P2 **Pad type** - VDDIO	
VSS	E5	--	0	0		**IO Supply** - VDD_P2 **Pad type** - VSSIO	
P2_0	H2	20	14	14	**ALT0** - P2_0 **ALT1** - TRIG_IN5 **ALT2** - FC9_P6 **ALT3** - uSDHC0_D5 **ALT4** - SCT0_IN0 **ALT5** - PWM1_A3 **ALT6** - FLEXIO0_D8 **ALT7** - SMARTDMA_PIO20 **ALT8** - FLEXSPI0_B_SS1_b **ALT10** - SAI0_RX_BCLK **ALT11** - ESPI0_RST	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	
P2_1	H1	21	15	15	**ALT0** - P2_1 **ALT1** - TRACE_CLK **ALT3** - uSDHC0_D4 **ALT4** - SCT0_IN1 **ALT5** - PWM1_B3 **ALT6** - FLEXIO0_D9 **ALT7** - SMARTDMA_PIO21 **ALT8** - FLEXSPI0_B_DQS **ALT9** - SINC0_MCLK_OUT0 **ALT10** - SAI0_RX_FS **ALT11** - ESPI0_NOTIFY	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	
P2_2	H3	22	16	16	**ALT0** - P2_2 **ALT1** - CLKOUT **ALT2** - FC9_P3 **ALT3** - uSDHC0_D1 **ALT4** - SCT0_OUT0	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN16

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT5 - PWM1_A2 **ALT6** - FLEXIO0_D10 **ALT7** - SMARTDMA_PIO22 **ALT8** - FLEXSPI0_B_SS0_b **ALT9** - SINC0_MCLK0 **ALT10** - SAI0_TXD0 **ALT11** - ESPI0_CSn		
P2_3	J3	23	17	17	**ALT0** - P2_3 **ALT2** - FC9_P1 **ALT3** - uSDHC0_D0 **ALT4** - SCT0_OUT1 **ALT5** - PWM1_B2 **ALT6** - FLEXIO0_D11 **ALT7** - SMARTDMA_PIO23 **ALT8** - FLEXSPI0_B_SCLK **ALT9** - SINC0_MBIT0 **ALT10** - SAI0_RXD0 **ALT11** - ESPI0_CLK	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	
P2_4	K3	24	18	18	**ALT0** - P2_4 **ALT2** - FC9_P0 **ALT3** - uSDHC0_CLK **ALT4** - SCT0_OUT2 **ALT5** - PWM1_A1 **ALT6** - FLEXIO0_D12 **ALT7** - SMARTDMA_PIO24 **ALT8** - FLEXSPI0_B_DATA0 **ALT9** - SINC0_MCLK1 **ALT10** - SAI0_RXD1 **ALT11** - ESPI0_DATA0	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN17
P2_5	K1	25	19	19	**ALT0** - P2_5 **ALT1** - TRIG_OUT3 **ALT2** - FC9_P2	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT3 - uSDHC0_CMD **ALT4** - SCT0_OUT3 **ALT5** - PWM1_B1 **ALT6** - FLEXIO0_D13 **ALT7** - SMARTDMA_PIO25 **ALT8** - FLEXSPI0_B_DATA1 **ALT9** - SINC0_MBIT1 **ALT10** - SAI0_TXD1 **ALT11** - ESPI0_DATA1		
VSS	G2	26	0	0		**IO Supply** - VDD_P2 **Pad type** - VSSIO	
VDD_P2	L7	27	--	--		**IO Supply** - VDD_P2 **Pad type** - VDDIO	
P2_6	K2	28	20	20	**ALT0** - P2_6 **ALT1** - TRIG_IN4 **ALT2** - FC9_P4 **ALT3** - uSDHC0_D3 **ALT4** - SCT0_OUT4 **ALT5** - PWM1_A0 **ALT6** - FLEXIO0_D14 **ALT7** - SMARTDMA_PIO26 **ALT8** - FLEXSPI0_B_DATA2 **ALT9** - SINC0_MCLK2 **ALT10** - SAI0_TX_BCLK **ALT11** - ESPI0_DATA2	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	
P2_7	L2	29	21	21	**ALT0** - P2_7 **ALT1** - TRIG_IN5 **ALT2** - FC9_P5 **ALT3** - uSDHC0_D2 **ALT4** - SCT0_OUT5 **ALT5** - PWM1_B0 **ALT6** - FLEXIO0_D15	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT7 - SMARTDMA_PIO27 **ALT8** - FLEXSPI0_B_DATA3 **ALT9** - SINC0_MBIT2 **ALT10** - SAI0_TX_FS **ALT11** - ESPI0_DATA3		
P2_8	M2	30	--	--	**ALT0** - P2_8 **ALT1** - TRACE_DATA0 **ALT3** - uSDHC0_D7 **ALT4** - SCT0_IN2 **ALT5** - PWM1_X0 **ALT6** - FLEXIO0_D16 **ALT7** - SMARTDMA_PIO28 **ALT8** - FLEXSPI0_B_DATA4 **ALT9** - SINC0_MCLK3 **ALT10** - SAI1_TXD0	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	
P2_9	M1	31	--	--	**ALT0** - P2_9 **ALT1** - TRACE_DATA1 **ALT3** - uSDHC0_D6 **ALT4** - SCT0_IN3 **ALT5** - PWM1_X1 **ALT6** - FLEXIO0_D17 **ALT7** - SMARTDMA_PIO29 **ALT8** - FLEXSPI0_B_DATA5 **ALT9** - SINC0_MBIT3 **ALT10** - SAI1_RXD0	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	
P2_10	M3	32	--	--	**ALT0** - P2_10 **ALT1** - TRACE_DATA2 **ALT4** - SCT0_IN4 **ALT5** - PWM1_X2 **ALT6** - FLEXIO0_D18 **ALT7** - SMARTDMA_PIO31	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT8 - FLEXSPI0_B_DATA6 **ALT9** - SINC0_MCLK4 **ALT10** - SAI1_RXD1		
P2_11	N4	33	--	--	**ALT0** - P2_11 **ALT1** - TRACE_DATA3 **ALT4** - SCT0_IN5 **ALT5** - PWM1_X3 **ALT6** - FLEXIO0_D19 **ALT7** - SMARTDMA_PIO30 **ALT8** - FLEXSPI0_B_DATA7 **ALT9** - SINC0_MBIT4 **ALT10** - SAI1_TXD1	**IO Supply** - VDD_P2 **Pad type** - FAST **Default** - DIS	
VSS	H5	34	0	0		**IO Supply** - VDD_P2 **Pad type** - VSSIO	
VDD_P2	--	35	--	--		**IO Supply** - VDD_P2 **Pad type** - VDDIO	
VDD_P4	P4	36	--	--		**IO Supply** - VDD_P4 **Pad type** - VDDIO	
VSS_P4	P7	37	--	--		**IO Supply** - VDD_P4 **Pad type** - VSSIO	
P4_0	P1	38	22	22	**ALT0** - P4_0 **ALT1** - TRIG_IN6 **ALT2** - FC2_P0 **ALT4** - CT_INP16 **ALT7** - SMARTDMA_PIO24 **ALT8** - PLU_IN0 **ALT9** - SINC0_MCLK3	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**VDD SYS** - WUU0_IN18
ANA_0	P3	39	22	22		**IO Supply** - VDD_P4 **Pad type** - ANA	**ANALOG** - ADC0_A0
ANA_1	R3	40	23	23		**IO Supply** - VDD_P4 **Pad type** - ANA	**ANALOG** - ADC0_B0

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
P4_1	P2	41	23	23	**ALT0** - P4_1 **ALT1** - TRIG_IN7 **ALT2** - FC2_P1 **ALT4** - CT_INP17 **ALT7** - SMARTDMA_PIO25 **ALT8** - PLU_IN1	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	
P4_2	T1	42	24	24	**ALT0** - P4_2 **ALT1** - TRIG_IN6 **ALT2** - FC2_P2 **ALT4** - CT_INP12 **ALT7** - SMARTDMA_PIO26 **ALT8** - PLU_IN2 **ALT9** - SINC0_MBIT3	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - DAC0_OUT/ADC0_A4/ ADC1_A4/CMP0_IN4N/ CMP1_IN4N/CMP2_IN4N
P4_3	U1	43	25	25	**ALT0** - P4_3 **ALT1** - TRIG_IN7 **ALT2** - FC2_P3 **ALT4** - CT_INP13 **ALT7** - SMARTDMA_PIO27 **ALT8** - PLU_IN3	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - DAC1_OUT/ADC0_B4/ ADC1_B4/CMP0_IN5N/ CMP1_IN5N/CMP2_IN5N **VDD SYS** - WUU0_IN19
P4_4	M6	44	26	26	**ALT0** - P4_4 **ALT2** - FC2_P4 **ALT4** - CT_INP14 **ALT7** - SMARTDMA_PIO28 **ALT8** - PLU_IN4 **ALT9** - SINC0_MCLK4	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	
ANA_4	T2	45	26	26		**IO Supply** - VDD_P4 **Pad type** - ANA	**ANALOG** - ADC1_A0
ANA_5	T3	46	27	27		**IO Supply** - VDD_P4 **Pad type** - ANA	**ANALOG** - ADC1_B0
P4_5	M8	47	27	27	**ALT0** - P4_5 **ALT2** - FC2_P5 **ALT4** - CT_INP15	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT7 - SMARTDMA_PIO29 **ALT8** - PLU_IN5 **ALT9** - SINC0_MBIT4		
ANA_6	U2	48	28	28		**IO Supply** - VDD_P4 **Pad type** - ANA	**ANALOG** - DAC2_OUT/ ADC0_A3/ADC1_A3
P4_6	N7	49	28	28	**ALT0** - P4_6 **ALT1** - TRIG_OUT4 **ALT2** - FC2_P6 **ALT4** - CT_INP18 **ALT7** - SMARTDMA_PIO30 **ALT8** - PLU_CLK	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	
P4_7	T4	50	29	29	**ALT0** - P4_7 **ALT4** - CT_INP19 **ALT7** - SMARTDMA_PIO31	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	
ANA_7	U4	51	29	29		**IO Supply** - VDD_P4 **Pad type** - ANA	**ANALOG** - VREFI/VREFO/ ADC0_A7/ADC1_A7
VDD_ANA	R4	52	30	30		**IO Supply** - VDD_P4 **Pad type** - VDDINT_3V	
VREFH	R5	53	31	31		**IO Supply** - VDD_P4 **Pad type** - ANA	**ANALOG** - VREFH
VREFL	R6	54	32	32		**IO Supply** - VDD_P4 **Pad type** - VSSINT	**ANALOG** - VREFL
VSS_P4	P6	55	33	33		**IO Supply** - VDD_P4 **Pad type** - VSSIO	
VDD_P4	N5	56	34	34		**IO Supply** - VDD_P4 **Pad type** - VDDIO	
P4_12	T6	57	35	35	**ALT0** - P4_12 **ALT1** - USB0_VBUS_DET **ALT2** - FC2_P0 **ALT4** - CT4_MAT0 **ALT6** - FLEXIO0_D20	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ISP** - USB0_VBUS_DET **ANALOG** - OPAMP0_INP0/ ADC0_A5/ADC1_A5 **VDD SYS** - WUU0_IN20

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT8 - PLU_OUT0 **ALT9** - SINC0_MCLK0 **ALT11** - CAN0_RXD		
P4_13	T7	58	36	36	**ALT0** - P4_13 **ALT1** - TRIG_IN8 **ALT2** - FC2_P1 **ALT3** - USB1_OTGn_ID **ALT4** - CT4_MAT1 **ALT6** - FLEXIO0_D21 **ALT8** - PLU_OUT1 **ALT9** - SINC0_MBIT0 **ALT11** - CAN0_TXD	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP0_INP1/ ADC0_B5/ADC1_B5
ANA_14	U6	59	36	36		**IO Supply** - VDD_P4 **Pad type** - LOLK	**ANALOG** - OPAMP0_INN
P4_14	N8	60	--	--	**ALT0** - P4_14 **ALT4** - CT4_MAT2 **ALT6** - FLEXIO0_D22 **ALT8** - PLU_OUT2	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	
P4_15	T8	61	37	37	**ALT0** - P4_15 **ALT1** - TRIG_OUT4 **ALT3** - USB1_VBUSVALID_EXT **ALT4** - CT4_MAT3 **ALT6** - FLEXIO0_D23 **ALT8** - PLU_OUT3 **ALT9** - SINC0_MCLK_OUT0 **ALT11** - CAN1_RXD	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP0_OUT/ ADC0_A1/CMP0_IN4P **VDD SYS** - WUU0_IN21
P4_16	R8	62	38	38	**ALT0** - P4_16 **ALT2** - FC2_P2 **ALT3** - USB1_OTGn_PWR **ALT4** - CT3_MAT0 **ALT6** - FLEXIO0_D24 **ALT8** - PLU_OUT4 **ALT9** - SINC0_MCLK1	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP1_INP0/ ADC0_A6

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT11 - CAN1_TXD		
P4_17	R9	63	39	39	**ALT0** - P4_17 **ALT1** - TRIG_IN9 **ALT2** - FC2_P3 **ALT3** - USB1_OTGn_OC **ALT4** - CT3_MAT1 **ALT6** - FLEXIO0_D25 **ALT8** - PLU_OUT5 **ALT9** - SINC0_MBIT1	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP1_INP1/ ADC0_B6
ANA_18	U8	64	39	39		**IO Supply** - VDD_P4 **Pad type** - LOLK	**ANALOG** - OPAMP1_INN
P4_18	N10	65	--	--	**ALT0** - P4_18 **ALT4** - CT3_MAT2 **ALT6** - FLEXIO0_D26 **ALT8** - PLU_OUT6	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	
P4_19	R10	66	40	--	**ALT0** - P4_19 **ALT1** - TRIG_OUT5 **ALT4** - CT3_MAT3 **ALT6** - FLEXIO0_D27 **ALT8** - PLU_OUT7 **ALT9** - SINC0_MCLK_OUT1	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP1_OUT/ ADC0_B1/CMP1_IN4P
P4_20	T10	67	41	--	**ALT0** - P4_20 **ALT1** - TRIG_IN8 **ALT2** - FC2_P4 **ALT4** - CT2_MAT0 **ALT6** - FLEXIO0_D28 **ALT9** - SINC0_MCLK2	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP2_INP0/ ADC1_A6
P4_21	T11	68	42	--	**ALT0** - P4_21 **ALT1** - TRIG_IN9 **ALT2** - FC2_P5 **ALT4** - CT2_MAT1 **ALT6** - FLEXIO0_D29 **ALT9** - SINC0_MBIT2	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP2_INP1/ ADC1_B6
ANA_22	U10	69	42	--		**IO Supply** - VDD_P4	**ANALOG** - OPAMP2_INN

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
						Pad type - LOLK	
P4_22	T12	70	--	--	**ALT0** - P4_22 **ALT4** - CT2_MAT2 **ALT6** - FLEXIO0_D30	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	
P4_23	U12	71	43	--	**ALT0** - P4_23 **ALT1** - TRIG_OUT5 **ALT2** - FC2_P6 **ALT4** - CT2_MAT3 **ALT6** - FLEXIO0_D31 **ALT9** - SINC0_MCLK_OUT2	**IO Supply** - VDD_P4 **Pad type** - SLOW **Default** - DIS	**ANALOG** - OPAMP2_OUT/ ADC0_A2/ADC0_B2/ ADC1_B3/CMP2_IN4P
VSS_P4	P9	72	--	--		**IO Supply** - VDD_P4 **Pad type** - VSSIO	
VDD_P4	--	73	--	--		**IO Supply** - VDD_P4 **Pad type** - VDDIO	
VSS	J4	--	--	0		**IO Supply** - VDD_USB **Pad type** - VSSIO	
USB1_DP	R13	74	--	40		**IO Supply** - VDD_USB **Pad type** - ANA	**ANALOG** - USB1_DP
USB1_DM	R14	75	--	41		**IO Supply** - VDD_USB **Pad type** - ANA	**ANALOG** - USB1_DM
USB1_ID	P11	--	--	--		**IO Supply** - VDD_USB **Pad type** - ANA	**ANALOG** - USB1_ID
USB1_VBUS	U14	76	--	42		**IO Supply** - VDD_USB **Pad type** - VDDINT_5V	**ANALOG** - USB1_VBUS
VSS	J8	77	--	43		**IO Supply** - VDD_USB **Pad type** - VSSIO	
VDD_USB	R12	78	44	44		**IO Supply** - VDD_USB **Pad type** - VDDIO	
USB0_DM	T14	79	45	45		**IO Supply** - VDD_USB **Pad type** - ANA	**ANALOG** - USB0_DM **VDD SYS** - WUU0_IN28
USB0_DP	T15	80	46	46		**IO Supply** - VDD_USB **Pad type** - ANA	**ANALOG** - USB0_DP **VDD SYS** - WUU0_IN29

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
VSS	T16	81	0	0		**IO Supply** - VDD_BAT **Pad type** - VSSIO	
VDD_BAT	T17	82	47	47		**IO Supply** - VDD_BAT **Pad type** - VDDIO	
P5_0	U16	83	48	48	**ALT0** - P5_0 **ALT1** - TRIG_IN10 **ALT2** - LPTMR0_ALT2	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - EXTAL32K/ADC1_B8
P5_1	U17	84	49	49	**ALT0** - P5_1 **ALT1** - TRIG_OUT6 **ALT2** - LPTMR1_ALT2	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - XTAL32K/ADC1_B9
P5_2	M10	85	50	50	**ALT0** - P5_2 **ALT1** - VBAT_WAKEUP_b **ALT2** - SPC_LPREQ **ALT3** - TAMPER0	**IO Supply** - VDD_BAT **Pad type** - RST **Default** - ALT1	**ANALOG** - ADC1_B10
P5_3	N11	86	51	51	**ALT0** - P5_3 **ALT1** - TRIG_IN11 **ALT2** - RTC_CLKOUT **ALT3** - TAMPER1	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - ADC1_B11
P5_4	M12	87	--	--	**ALT0** - P5_4 **ALT1** - TRIG_OUT7 **ALT2** - SPC_LPREQ **ALT3** - TAMPER2	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - ADC1_B12
P5_5	K12	88	--	--	**ALT0** - P5_5 **ALT1** - TRIG_IN10 **ALT2** - LPTMR0_ALT2 **ALT3** - TAMPER3	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - ADC1_B13
P5_6	K13	89	--	--	**ALT0** - P5_6 **ALT1** - TRIG_OUT6 **ALT2** - LPTMR1_ALT2 **ALT3** - TAMPER4	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - ADC1_B14
P5_7	L13	90	--	--	**ALT0** - P5_7 **ALT1** - TRIG_IN11 **ALT3** - TAMPER5	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - ADC1_B15

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
P5_8	L14	91	--	--	**ALT0** - P5_8 **ALT1** - TRIG_OUT7 **ALT3** - TAMPER6	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - ADC1_B16
P5_9	M14	92	--	--	**ALT0** - P5_9 **ALT3** - TAMPER7	**IO Supply** - VDD_BAT **Pad type** - AON **Default** - DIS	**ANALOG** - ADC1_B17
VSS	E13	--	--	--		**IO Supply** - VDD_BAT **Pad type** - VSSIO	
VSS_DCDC	P16	93,94	52	52		**IO Supply** - VDD_DCDC **Pad type** - VSSIO	
DCDC_LX	P17	95,96	53	53		**IO Supply** - VDD_DCDC **Pad type** - ANA	**ANALOG** - DCDC_LX
VDD_DCDC	R15	97	54	54		**IO Supply** - VDD_DCDC **Pad type** - VDDIO	
VDD_LDO_SYS	P15	98	54	54		**IO Supply** - VDD_P3 **Pad type** - VDDIO	
VDD_SYS	N14	99	55	55		**IO Supply** - VDD_P3 **Pad type** - VDDINT	
VSS	J14	--	0	0		**IO Supply** - VDD_P3 **Pad type** - VSSIO	
P3_23	M15	100	--	--	**ALT0** - P3_23 **ALT3** - FC6_P3 **ALT4** - CT_INP11 **ALT5** - PWM1_X3 **ALT6** - FLEXIO0_D31 **ALT7** - SMARTDMA_PIO23 **ALT10** - SAI1_TXD1	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_22	M16	101	--	--	**ALT0** - P3_22 **ALT2** - FC8_P6 **ALT3** - FC6_P2 **ALT4** - CT_INP10 **ALT5** - PWM1_X2 **ALT6** - FLEXIO0_D30	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT7 - SMARTDMA_PIO22 **ALT9** - SIM0_VCCEN **ALT10** - SAI1_RXD1		
P3_21	L16	102	56	56	**ALT0** - P3_21 **ALT1** - TRIG_OUT1 **ALT2** - FC8_P5 **ALT3** - FC6_P1 **ALT4** - CT2_MAT3 **ALT5** - PWM1_B3 **ALT6** - FLEXIO0_D29 **ALT7** - SMARTDMA_PIO21 **ALT9** - SIM0_RST **ALT10** - SAI1_RXD0 **ALT11** - PF_SPI_CS1_DIS_n	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_20	M17	103	57	57	**ALT0** - P3_20 **ALT1** - TRIG_OUT0 **ALT2** - FC8_P4 **ALT3** - FC6_P0 **ALT4** - CT2_MAT2 **ALT5** - PWM1_A3 **ALT6** - FLEXIO0_D28 **ALT7** - SMARTDMA_PIO20 **ALT9** - SIM0_PD **ALT10** - SAI1_TXD0 **ALT11** - PF_SPI_CS0_DIS_n	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN27
P3_19	K17	--	--	--	**ALT0** - P3_19 **ALT2** - FC7_P6 **ALT4** - CT2_MAT1 **ALT5** - PWM1_X1 **ALT6** - FLEXIO0_D27 **ALT7** - SMARTDMA_PIO19 **ALT10** - SAI1_RX_FS	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
VSS	K9	104	0	0		**IO Supply** - VDD_P3 **Pad type** - VSSIO	
VDD_CORE	L11	105	58	58		**IO Supply** - VDD_P3 **Pad type** - VDDINT	
VDD_P3	G11	106	59	59		**IO Supply** - VDD_P3 **Pad type** - VDDIO	
P3_18	K16	--	--	--	**ALT0** - P3_18 **ALT3** - FC6_P6 **ALT4** - CT2_MAT0 **ALT5** - PWM1_X0 **ALT6** - FLEXIO0_D26 **ALT7** - SMARTDMA_PIO18 **ALT10** - SAI1_RX_BCLK	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_17	K15	107	60	60	**ALT0** - P3_17 **ALT2** - FC8_P3 **ALT4** - CT_INP9 **ALT5** - PWM1_B2 **ALT6** - FLEXIO0_D25 **ALT7** - SMARTDMA_PIO17 **ALT9** - SIM0_IO **ALT10** - SAI1_TX_FS	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN26
P3_16	J15	108	61	61	**ALT0** - P3_16 **ALT2** - FC8_P2 **ALT4** - CT_INP8 **ALT5** - PWM1_A2 **ALT6** - FLEXIO0_D24 **ALT7** - SMARTDMA_PIO16 **ALT9** - SIM0_CLK **ALT10** - SAI1_TX_BCLK	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_15	H15	109	62	62	**ALT0** - P3_15 **ALT2** - FC8_P1 **ALT4** - CT_INP7 **ALT5** - PWM1_B1	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT6 - FLEXIO0_D23 **ALT7** - SMARTDMA_PIO15 **ALT8** - FLEXSPI0_A_DATA7 **ALT10** - SAI0_RX_FS **ALT11** - PF_SPI_SCKIN		
P3_14	H17	110	63	63	**ALT0** - P3_14 **ALT2** - FC8_P0 **ALT4** - CT_INP6 **ALT5** - PWM1_A1 **ALT6** - FLEXIO0_D22 **ALT7** - SMARTDMA_PIO14 **ALT8** - FLEXSPI0_A_DATA6 **ALT10** - SAI0_RX_BCLK **ALT11** - PF_SPI_DATA	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN25
P3_13	H16	111	64	64	**ALT0** - P3_13 **ALT2** - FC7_P5 **ALT3** - FC6_P5 **ALT4** - CT1_MAT3 **ALT5** - PWM1_B0 **ALT6** - FLEXIO0_D21 **ALT7** - SMARTDMA_PIO13 **ALT8** - FLEXSPI0_A_DATA5 **ALT10** - SAI0_TXD1 **ALT11** - PF_SPI_CS0_n	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_12	G16	112	65	65	**ALT0** - P3_12 **ALT2** - FC7_P4 **ALT3** - FC6_P4 **ALT4** - CT1_MAT2 **ALT5** - PWM1_A0 **ALT6** - FLEXIO0_D20 **ALT7** - SMARTDMA_PIO12	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT8 - FLEXSPI0_A_DATA4 **ALT10** - SAI0_RXD1		
VSS	N13	113	0	0		**IO Supply** - VDD_P3 **Pad type** - VSSIO	
VDD_P3	H10	114	66	66		**IO Supply** - VDD_P3 **Pad type** - VDDIO	
P3_11	F16	115	67	67	**ALT0** - P3_11 **ALT2** - FC6_P3 **ALT3** - FC7_P5 **ALT4** - CT1_MAT1 **ALT5** - PWM0_B3 **ALT6** - FLEXIO0_D19 **ALT7** - SMARTDMA_PIO11 **ALT8** - FLEXSPI0_A_DATA3 **ALT9** - SIM0_IO **ALT10** - SAI0_RXD0 **ALT11** - PF_QSPI_DATA3	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN24
P3_10	F17	116	68	68	**ALT0** - P3_10 **ALT2** - FC6_P2 **ALT3** - FC7_P4 **ALT4** - CT1_MAT0 **ALT5** - PWM0_A3 **ALT6** - FLEXIO0_D18 **ALT7** - SMARTDMA_PIO10 **ALT8** - FLEXSPI0_A_DATA2 **ALT9** - SIM0_CLK **ALT10** - SAI0_TXD0 **ALT11** - PF_QSPI_DATA2	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_9	F15	117	69	69	**ALT0** - P3_9 **ALT2** - FC6_P5	**IO Supply** - VDD_P3 **Pad type** - FAST	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT3 - FC7_P2 **ALT4** - CT_INP5 **ALT5** - PWM0_B2 **ALT6** - FLEXIO0_D17 **ALT7** - SMARTDMA_PIO9 **ALT8** - FLEXSPI0_A_DATA1 **ALT9** - SIM0_RST **ALT10** - SAI0_TX_FS **ALT11** - PF_QSPI_DATA1	**Default** - DIS	
P3_8	E14	118	70	70	**ALT0** - P3_8 **ALT2** - FC6_P4 **ALT3** - FC7_P0 **ALT4** - CT_INP4 **ALT5** - PWM0_A2 **ALT6** - FLEXIO0_D16 **ALT7** - SMARTDMA_PIO8 **ALT8** - FLEXSPI0_A_DATA0 **ALT9** - SIM0_PD **ALT10** - SAI0_TX_BCLK **ALT11** - PF_QSPI_DATA0	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN23
P3_7	D14	119	71	71	**ALT0** - P3_7 **ALT2** - FC6_P6 **ALT3** - FC7_P1 **ALT4** - CT4_MAT3 **ALT5** - PWM0_B1 **ALT6** - FLEXIO0_D15 **ALT7** - SMARTDMA_PIO7 **ALT8** - FLEXSPI0_A_SCLK **ALT9** - SIM0_VCCEN **ALT10** - SAI0_MCLK	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT11 - PF_QSPI_SCKIN		
P3_6	D17	120	72	72	**ALT0** - P3_6 **ALT1** - CLKOUT **ALT2** - FC6_P1 **ALT4** - CT4_MAT2 **ALT5** - PWM0_A1 **ALT6** - FLEXIO0_D14 **ALT7** - SMARTDMA_PIO6 **ALT8** - FLEXSPI0_A_DQS **ALT9** - SIM1_VCCEN **ALT10** - SAI1_MCLK **ALT11** - PF_QSPI_CS_n	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
VSS	P12	121	0	0		**IO Supply** - VDD_P3 **Pad type** - VSSIO	
VDD_P3	H12	122	73	73		**IO Supply** - VDD_P3 **Pad type** - VDDIO	
P3_5	G14	123	--	--	**ALT0** - P3_5 **ALT2** - FC7_P3 **ALT4** - CT_INP19 **ALT5** - PWM0_X3 **ALT6** - FLEXIO0_D13 **ALT7** - SMARTDMA_PIO5 **ALT9** - SIM1_IO	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_4	F14	124	--	--	**ALT0** - P3_4 **ALT2** - FC7_P2 **ALT4** - CT_INP18 **ALT5** - PWM0_X2 **ALT6** - FLEXIO0_D12 **ALT7** - SMARTDMA_PIO4 **ALT9** - SIM1_CLK	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_3	D16	125	--	--	**ALT0** - P3_3 **ALT2** - FC7_P1	**IO Supply** - VDD_P3 **Pad type** - FAST	

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT4 - CT4_MAT1 **ALT5** - PWM0_X1 **ALT6** - FLEXIO0_D11 **ALT7** - SMARTDMA_PIO3 **ALT9** - SIM1_RST	**Default** - DIS	
P3_2	D15	126	--	--	**ALT0** - P3_2 **ALT2** - FC7_P0 **ALT4** - CT4_MAT0 **ALT5** - PWM0_X0 **ALT6** - FLEXIO0_D10 **ALT7** - SMARTDMA_PIO2 **ALT9** - SIM1_PD	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_1	C15	127	74	74	**ALT0** - P3_1 **ALT1** - TRIG_IN1 **ALT2** - FC6_P0 **ALT3** - FC7_P6 **ALT4** - CT_INP17 **ALT5** - PWM0_B0 **ALT6** - FLEXIO0_D9 **ALT7** - SMARTDMA_PIO1 **ALT8** - FLEXSPI0_A_SS1_b **ALT11** - PF_QSPI_CS1_DIS	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	
P3_0	B17	128	75	75	**ALT0** - P3_0 **ALT1** - TRIG_IN0 **ALT3** - FC7_P3 **ALT4** - CT_INP16 **ALT5** - PWM0_A0 **ALT6** - FLEXIO0_D8 **ALT7** - SMARTDMA_PIO0 **ALT8** - FLEXSPI0_A_SS0_b	**IO Supply** - VDD_P3 **Pad type** - FAST **Default** - DIS	**VDD SYS** - WUU0_IN22

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT11 - PF_QSPI_CS0_DIS		
VSS	H9	--	--	--		**IO Supply** - VDD_P3 **Pad type** - VSSIO	
VDD_P3	--	129	--	--		**IO Supply** - VDD_P3 **Pad type** - VDDIO	
VDD	--	130	--	--		**IO Supply** - VDD **Pad type** - VDDIO	
VSS	J10	131	--	--		**IO Supply** - VDD **Pad type** - VSSIO	
P0_0	A17	132	76	76	**ALT0** - P0_0 **ALT1** - TMS/SWDIO **ALT2** - FC1_P0 **ALT4** - CT_INP0	**IO Supply** - VDD **Pad type** - MED **Default** - ALT1	
P0_1	A16	133	77	77	**ALT0** - P0_1 **ALT1** - TCLK/SWCLK **ALT2** - FC1_P1 **ALT4** - CT_INP1	**IO Supply** - VDD **Pad type** - MED **Default** - ALT1	
P0_2	B16	134	78	78	**ALT0** - P0_2 **ALT1** - TDO/SWO **ALT2** - FC1_P2 **ALT4** - CT0_MAT0 **ALT5** - UTICK_CAP0 **ALT10** - I3C0_PUR	**IO Supply** - VDD **Pad type** - MED **Default** - ALT1	
P0_3	B15	135	79	79	**ALT0** - P0_3 **ALT1** - TDI **ALT2** - FC1_P3 **ALT4** - CT0_MAT1 **ALT5** - UTICK_CAP1 **ALT8** - HSCMP0_OUT	**IO Supply** - VDD **Pad type** - MED **Default** - ALT1	**ANALOG** - CMP1_IN1
P0_4	B14	136	80	80	**ALT0** - P0_4 **ALT1** - EWM0_IN **ALT2** - FC0_P0 **ALT3** - FC1_P4 **ALT4** - CT0_MAT2	**IO Supply** - VDD **Pad type** - MED+I2C **Default** - DIS	**ANALOG** - TSI0_CH8 **VDD SYS** - WUU0_IN0

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT5 - UTICK_CAP2 **ALT8** - HSCMP1_OUT **ALT9** - PDM0_CLK		
P0_5	A14	137	81	81	**ALT0** - P0_5 **ALT1** - EWM0_OUT_b **ALT2** - FC0_P1 **ALT3** - FC1_P5 **ALT4** - CT0_MAT3 **ALT5** - UTICK_CAP3 **ALT9** - PDM0_DATA0	**IO Supply** - VDD **Pad type** - MED+I2C **Default** - DIS	**ANALOG** - TSI0_CH9
P0_6	C14	138	82	82	**ALT0** - P0_6 **ALT1** - ISPMODE_N **ALT2** - FC0_P2 **ALT3** - FC1_P6 **ALT4** - CT_INP2 **ALT8** - HSCMP2_OUT **ALT9** - PDM0_DATA1	**IO Supply** - VDD **Pad type** - MED **Default** - ALT1	**ISP** - ISPMODE_N **ANALOG** - TSI0_CH10
P0_7	C13	139	--	--	**ALT0** - P0_7 **ALT2** - FC0_P3 **ALT4** - CT_INP3	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - CMP2_IN1 **VDD SYS** - WUU0_IN1
P0_8	C12	140	--	--	**ALT0** - P0_8 **ALT2** - FC0_P4 **ALT4** - CT_INP0 **ALT6** - FLEXIO0_D0	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B8
P0_9	A12	141	--	--	**ALT0** - P0_9 **ALT2** - FC0_P5 **ALT4** - CT_INP1 **ALT6** - FLEXIO0_D1	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B9
P0_10	B12	142	--	--	**ALT0** - P0_10 **ALT2** - FC0_P6 **ALT4** - CT0_MAT0 **ALT6** - FLEXIO0_D2	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B10
P0_11	B11	143	--	--	**ALT0** - P0_11 **ALT4** - CT0_MAT1 **ALT6** - FLEXIO0_D3	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B11

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT8 - HSCMP2_OUT		
VDD	G7	144	83	83		**IO Supply** - VDD **Pad type** - VDDIO	
VSS	D9	145	0	0		**IO Supply** - VDD **Pad type** - VSSIO	
P0_12	D11	146	--	--	**ALT0** - P0_12 **ALT2** - FC1_P4 **ALT3** - FC0_P0 **ALT4** - CT0_MAT2 **ALT6** - FLEXIO0_D4	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B12/NVM_TM0
P0_13	F12	147	--	--	**ALT0** - P0_13 **ALT2** - FC1_P5 **ALT3** - FC0_P1 **ALT4** - CT0_MAT3 **ALT6** - FLEXIO0_D5	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B13/NVM_TM1
P0_14	E11	148	--	--	**ALT0** - P0_14 **ALT2** - FC1_P6 **ALT3** - FC0_P2 **ALT4** - CT_INP2 **ALT5** - UTICK_CAP0 **ALT6** - FLEXIO0_D6	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B14/NVM_TM2
P0_15	G13	149	--	--	**ALT0** - P0_15 **ALT3** - FC0_P3 **ALT4** - CT_INP3 **ALT5** - UTICK_CAP1 **ALT6** - FLEXIO0_D7	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B15/NVM_TM3
P0_16	B10	150	84	84	**ALT0** - P0_16 **ALT2** - FC0_P0 **ALT4** - CT0_MAT0 **ALT5** - UTICK_CAP2 **ALT6** - FLEXIO0_D0 **ALT9** - PDM0_CLK **ALT10** - I3C0_SDA	**IO Supply** - VDD **Pad type** - MED+I2C+I3C **Default** - DIS	**ISP** - I2C_SDA **ANALOG** - TSI0_CH11/ADC0_A8 **VDD SYS** - WUU0_IN2
P0_17	A10	151	85	85	**ALT0** - P0_17 **ALT2** - FC0_P1	**IO Supply** - VDD **Pad type** - MED+I2C	**ISP** - I2C_SCL

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT4 - CT0_MAT1 **ALT5** - UTICK_CAP3 **ALT6** - FLEXIO0_D1 **ALT9** - PDM0_DATA0 **ALT10** - I3C0_SCL	**Default** - DIS	**ANALOG** - TSI0_CH12/ADC0_A9
P0_18	C10	152	86	86	**ALT0** - P0_18 **ALT1** - EWM0_IN **ALT2** - FC0_P2 **ALT4** - CT0_MAT2 **ALT6** - FLEXIO0_D2 **ALT8** - HSCMP0_OUT **ALT9** - PDM0_DATA1	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH13/ADC0_A10
P0_19	C9	153	87	87	**ALT0** - P0_19 **ALT1** - EWM0_OUT_b **ALT2** - FC0_P3 **ALT4** - CT0_MAT3 **ALT6** - FLEXIO0_D3 **ALT8** - HSCMP1_OUT	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH14/ADC0_A11 **VDD SYS** - WUU0_IN3
P0_20	C8	154	88	88	**ALT0** - P0_20 **ALT2** - FC0_P4 **ALT3** - FC1_P0 **ALT4** - CT_INP0 **ALT6** - FLEXIO0_D4 **ALT10** - I3C0_SDA	**IO Supply** - VDD **Pad type** - MED+I2C+I3C **Default** - DIS	**ANALOG** - TSI0_CH15/ADC0_A12 **VDD SYS** - WUU0_IN4
P0_21	A8	155	89	89	**ALT0** - P0_21 **ALT2** - FC0_P5 **ALT3** - FC1_P1 **ALT4** - CT_INP1 **ALT6** - FLEXIO0_D5 **ALT10** - I3C0_SCL	**IO Supply** - VDD **Pad type** - MED+I2C **Default** - DIS	**ANALOG** - TSI0_CH16/ADC0_A13
P0_22	B8	156	90	90	**ALT0** - P0_22 **ALT1** - EWM0_IN **ALT2** - FC0_P6 **ALT3** - FC1_P2 **ALT4** - CT_INP2 **ALT6** - FLEXIO0_D6	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_A14/CMP1_IN2

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT10 - I3C0_PUR		
P0_23	B7	157	91	91	**ALT0** - P0_23 **ALT1** - EWM0_OUT_b **ALT3** - FC1_P3 **ALT4** - CT_INP3 **ALT6** - FLEXIO0_D7	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_A15/CMP2_IN2 **VDD SYS** - WUU0_IN5
VSS	H13	--	--	--		**IO Supply** - VDD **Pad type** - VSSIO	
P0_24	B6	158	--	--	**ALT0** - P0_24 **ALT2** - FC1_P0 **ALT4** - CT0_MAT0	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B16
P0_25	A6	159	--	--	**ALT0** - P0_25 **ALT2** - FC1_P1 **ALT4** - CT0_MAT1	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B17
P0_26	F10	160	--	--	**ALT0** - P0_26 **ALT2** - FC1_P2 **ALT4** - CT0_MAT2	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B18
P0_27	E10	161	--	--	**ALT0** - P0_27 **ALT2** - FC1_P3 **ALT4** - CT0_MAT3	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B19
P0_28	E8	--	--	--	**ALT0** - P0_28 **ALT2** - FC1_P4 **ALT3** - FC0_P4 **ALT4** - CT_INP0	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B20
P0_29	F8	--	--	--	**ALT0** - P0_29 **ALT2** - FC1_P5 **ALT3** - FC0_P5 **ALT4** - CT_INP1	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B21
P0_30	E7	--	--	--	**ALT0** - P0_30 **ALT2** - FC1_P6 **ALT3** - FC0_P6 **ALT4** - CT_INP2	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - ADC0_B22
P0_31	D7	--	--	--	**ALT0** - P0_31 **ALT4** - CT_INP3	**IO Supply** - VDD **Pad type** - MED	**ANALOG** - ADC0_B23

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
						Default - DIS	
P1_0	C6	162	92	92	**ALT0** - P1_0 **ALT1** - TRIG_IN0 **ALT2** - FC3_P0 **ALT3** - FC4_P4 **ALT4** - CT_INP4 **ALT5** - SCT0_OUT6 **ALT6** - FLEXIO0_D8 **ALT10** - SAI1_TX_BCLK	**IO Supply** - VDD **Pad type** - MED+I2C **Default** - DIS	**ISP** - SPI_SDO **ANALOG** - TSI0_CH0/ ADC0_A16/CMP0_IN0 **VDD SYS** - WUU0_IN6/ LPTMR0_ALT3
P1_1	C5	163	93	93	**ALT0** - P1_1 **ALT1** - TRIG_IN1 **ALT2** - FC3_P1 **ALT3** - FC4_P5 **ALT4** - CT_INP5 **ALT5** - SCT0_OUT7 **ALT6** - FLEXIO0_D9 **ALT10** - SAI1_TX_FS	**IO Supply** - VDD **Pad type** - MED+I2C **Default** - DIS	**ISP** - SPI_SCK **ANALOG** - TSI0_CH1/ ADC0_A17/CMP1_IN0
P1_2	C4	164	94	94	**ALT0** - P1_2 **ALT1** - TRIG_OUT0 **ALT2** - FC3_P2 **ALT3** - FC4_P6 **ALT4** - CT1_MAT0 **ALT5** - SCT0_IN6 **ALT6** - FLEXIO0_D10 **ALT9** - ENET0_MDC **ALT10** - SAI1_TXD0 **ALT11** - CAN0_TXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ISP** - SPI_SDI **ANALOG** - TSI0_CH2/ ADC0_A18/CMP2_IN0
P1_3	B4	165	95	95	**ALT0** - P1_3 **ALT1** - TRIG_OUT1 **ALT2** - FC3_P3 **ALT4** - CT1_MAT1 **ALT5** - SCT0_IN7 **ALT6** - FLEXIO0_D11 **ALT9** - ENET0_MDIO **ALT10** - SAI1_RXD0 **ALT11** - CAN0_RXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ISP** - SPI_PCS **ANALOG** - TSI0_CH3/ ADC0_A19/CMP0_IN1 **VDD SYS** - WUU0_IN7

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
VDD	H6	166	96	96		**IO Supply** - VDD **Pad type** - VDDIO	
VSS	D12	167	0	0		**IO Supply** - VDD **Pad type** - VSSIO	
P1_4	A4	168	97	97	**ALT0** - P1_4 **ALT1** - FREQME_CLK_IN0 **ALT2** - FC3_P4 **ALT3** - FC5_P0 **ALT4** - CT1_MAT2 **ALT5** - SCT0_OUT0 **ALT6** - FLEXIO0_D12 **ALT7** - SMARTDMA_PIO0 **ALT9** - ENET0_TX_CLK **ALT10** - SAI0_TXD1	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH4/ ADC0_A20/CMP0_IN2 **VDD SYS** - WUU0_IN8
P1_5	B3	169	98	98	**ALT0** - P1_5 **ALT1** - FREQME_CLK_IN1 **ALT2** - FC3_P5 **ALT3** - FC5_P1 **ALT4** - CT1_MAT3 **ALT5** - SCT0_OUT1 **ALT6** - FLEXIO0_D13 **ALT7** - SMARTDMA_PIO1 **ALT9** - ENET0_TXEN **ALT10** - SAI0_RXD1	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH5/ ADC0_A21/CMP0_IN3
P1_6	B2	170	99	99	**ALT0** - P1_6 **ALT1** - TRIG_IN2 **ALT2** - FC3_P6 **ALT3** - FC5_P2 **ALT4** - CT_INP6 **ALT5** - SCT0_IN0 **ALT6** - FLEXIO0_D14 **ALT7** - SMARTDMA_PIO2	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH6/ADC0_A22

Table continues on the next page...

Table 119. pinmux (continued)

Pin Name	184BGA ALL	172HDQFP ALL	100HLQFP N94	100HLQFP N54, N53, N52, N24	Pinmux Assignment	Pad Settings	Alternate Functions
					ALT9 - ENET0_TXD0 **ALT10** - SAI1_RX_BCLK **ALT11** - CAN1_TXD		
P1_7	A2	171	100	100	**ALT0** - P1_7 **ALT1** - TRIG_OUT2 **ALT3** - FC5_P3 **ALT4** - CT_INP7 **ALT5** - SCT0_IN1 **ALT6** - FLEXIO0_D15 **ALT7** - SmartDMA_PIO3 **ALT8** - PLU_CLK **ALT9** - ENET0_TXD1 **ALT10** - SAI1_RX_FS **ALT11** - CAN1_RXD	**IO Supply** - VDD **Pad type** - MED **Default** - DIS	**ANALOG** - TSI0_CH7/ADC0_A23 **VDD SYS** - WUU0_IN9

NOTE

- For BGA package, all balls with same name are shorted together on BGA package.
- VSS_ANA and VSS_P4 are shorted together on package.
- +I3C in Pad Type means strong pull up resistor is implemented on the pin. PV bit is implemented in the Pin Control register of the pin.
- +I2C in Pad Type represents I2C filter is implemented on the pin. PFE bit is implemented in the Pin Control register of the pin.
- DIS in default column means the pin's input buffer is disabled by default.
- AON and RST pads support passive filter. PFE bit is implemented in the Pin Control register of the pin.
- PE, PS, SRE, and DSE are supported in the Pin Control register of all types of IO.

Chapter 13
ROM API

13.1 Overview

This section provides an introduction to the ROM API in this device.

NOTE

This chapter is also available in the *MCX N Security Reference Manual*, with differences related to security functionality. The chapter in the Reference Manual does not show the security relevant information in it.

13.1.1 Features

- The ROM API supports programming both the Programmable FLASH region (store application code and data) and the CFPA and CMPA Regions (store device configurations, boot configuration, in-field programmable data).
- The ROM API enables the Serial NOR FLASH programming via the FlexSPI Flash Driver API.
- The ROM API supports OTP-eFuse read and programming.
- The ROM API provides native support for Nboot API in the user application.
- The ROM API supports In-application-programming functionality by the IAP API, via the unified memory interface or the Sbloader API.

13.2 Functional description

This section introduces the ROM API structure first, then describes the FLASH API, OTP API, FLEXSPI NOR API, IAP API and NBOOT APIs. This section also demonstrates the typical usage of each API.

- The primary purpose of the FLASH API is to update the programmable FLASH area and specify the parameters in the CMPA and the CFPA.
- The FLEXSPI FLASH API eases the support of various Serial NOR devices in the market.
- OTP API provides the functionality to advance lifecycle; program/read critical one-time programmable parameters and some general-purpose one-time programmable FUSE field.
- IAP API enables the flexibility of doing in-application-programming by the memory interface or the dedicated SBloader API.

NOTE

All ROM APIs are callable from secure world only. Therefore, all function pointers use secure address aliases. Applications should switch to secure mode and then call ROM API or create NSC veneers for ROM APIs.

13.2.1 ROM API structure

The ROM API table locates at address 0x1303fc00. See Figure 54.

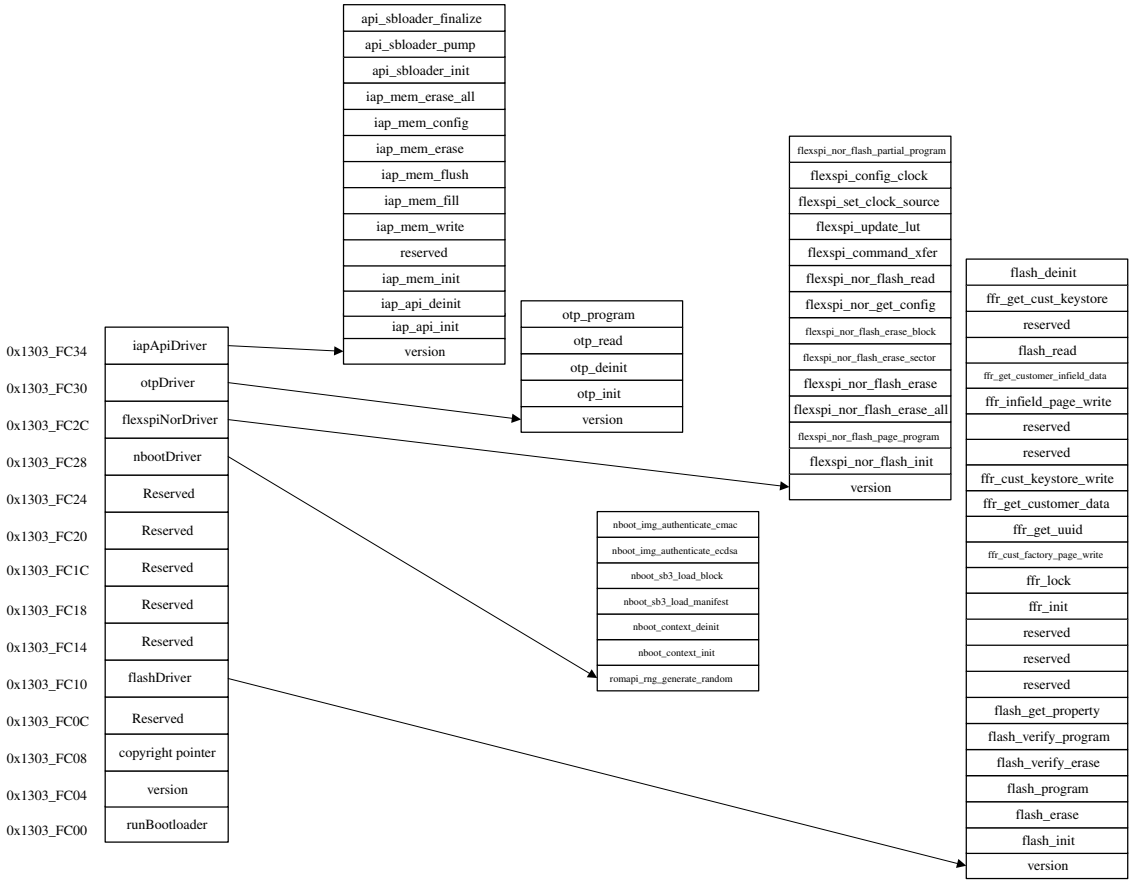


Figure 54. ROM API structure

13.2.2 FLASH APIs

The FLASH API set enables the following features:

- Initialize FLASH controller
- Erase and verify the specified FLASH area
- Program and verify the specified FLASH page
- Retrieve FLASH properties
- Initialize or Lock the CMPA and CFPA Regions
- Program and Read CMPA
- Program and Read CFPA

The FLASH APIs are organized in the FLASH Driver API Interface structure. See the API layout and API prototypes from the following data structure. The whole FLASH API wrapper is available in the SDK release package.

The bootloader API prototypes are:

```
typedef struct FLASHDriverInterface
{
    standard_version_t version; /* flash driver API version number */
```

```
    /* FLASH driver */
    status_t (*flash_init)(flash_config_t *config);
    status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t
key);
    status_t (*flash_program)(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t
lengthInBytes);
    status_t (*flash_verify_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes);
    status_t (*flash_verify_program)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes,
const uint8_t *expectedData, uint32_t *failedAddress, uint32_t *failedData);
    status_t (*flash_get_property)(flash_config_t *config, flash_property_tag_t whichProperty,
uint32_t *value);
    uint32_t reserved0[3];

    /* FLASH FFR driver */
    status_t (*ffr_init)(flash_config_t *config);
    status_t (*ffr_lock)(flash_config_t *config);
    status_t (*ffr_cust_factory_page_write)(flash_config_t *config, uint8_t *page_data, bool
seal_part);
    status_t (*ffr_get_uuid)(flash_config_t *config, uint8_t *uuid);
    status_t (*ffr_get_customer_data)(flash_config_t *config, uint8_t *pData, uint32_t offset,
uint32_t len);
    status_t (*ffr_cust_keystore_write)(flash_config_t *config, ffr_key_store_t *pKeyStore);
    status_t reserved1;
    status_t reserved2;
    status_t (*ffr_infield_page_write)(flash_config_t *config, uint8_t *page_data, uint32_t
valid_len);
    status_t (*ffr_get_customer_infield_data)(flash_config_t *config, uint8_t *pData, uint32_t
offset, uint32_t len);
    status_t (*flash_read)(flash_config_t *config, uint32_t start, uint8_t *dest, uint32_t
lengthInBytes);
    status_t reserved3;
    status_t (*flash_get_cust_keystore)(flash_config_t *config, uint8_t *pData, uint32_t offset,
uint32_t len);
    status_t (*flash_deinit)(flash_config_t *config);
}
flash_driver_interface_t;
```

Each FLASH API depends on a common FLASH context structure named *flash_config_t* to perform the proper FLASH operation. See the structure details below.

```

/*! @brief FLASH driver state information.
*
* An instance of this structure is allocated by the user of the flash driver and
* passed into each of the driver APIs.
*/
typedef struct
{
    uint32_t PFlashBlockBase;  /*!< A base address of the first PFlash block */
    uint32_t PFlashTotalSize;  /*!< The size of the combined PFlash block. */
    uint32_t PFlashBlockCount; /*!< A number of PFlash blocks. */
    uint32_t PFlashPageSize;   /*!< The size in bytes of a page of PFlash. */
    uint32_t PFlashSectorSize; /*!< The size in bytes of a sector of PFlash. */
    flash_ffr_config_t ffrConfig;
    uint32_t reserved0[5];
    uint32_t *nbootCtx;
    bool useAhbRead;
}
flash_config_t;
```

The flash_ffr_config_t is defined as below:

```
/*! @brief FLASH controller paramter config. */

typedef struct
{
    uint32_t ffrBlockBase;
    uint32_t ffrTotalSize;
    uint32_t ffrPageSize;
    uint32_t sectorSize;
    uint32_t cfpaPageVersion;
    uint32_t cfpaPageOffset;
}
flash_ffr_config_t;
```

flash_ffr_config_t is managed by the FFR_Init API call, the user application doesn't need to touch it.

The ffr_key_store_t and flash_ffr_config_t definitions are:

```
#define FLASH_FFR_KETBLOB_SIZE (0x30u)
typedef struct
{
    uint8_t reserved[1][FLASH_FFR_KETBLOB_SIZE];
}
ffr_key_store_t;
typedef enum
{
    kFLASH_PropertyPflashSectorSize = 0x00U, /*!< Pflash sector size property.*/
    kFLASH_PropertyPflashTotalSize = 0x01U, /*!< Pflash total size property.*/
    kFLASH_PropertyPflashBlockSize = 0x02U, /*!< Pflash block size property.*/
    kFLASH_PropertyPflashBlockCount = 0x03U, /*!< Pflash block count property.*/
    kFLASH_PropertyPflashBlockBaseAddr = 0x04U, /*!< Pflash block base address property.*/
    kFLASH_PropertyPflashPageSize = 0x30U, /*!< Pflash page size property.*/
    kFLASH_PropertyPflashSystemFreq = 0x31U, /*!< System Frequency property.*/
    kFLASH_PropertyFfrSectorSize = 0x40U, /*!< FFR sector size property.*/
    kFLASH_PropertyFfrTotalSize = 0x41U, /*!< FFR total size property.*/
    kFLASH_PropertyFfrBlockBaseAddr = 0x42U, /*!< FFR block base address property.*/
    kFLASH_PropertyFfrPageSize = 0x43U, /*!< FFR page size property.*/
}
flash_property_tag_t;
```

13.2.2.1 Version

The "version" field in the FLASH API table indicates the current FLASH API version in the ROM bootloader.

Prototype :

```
standard_version_t version;
```

Table 120. Definition of the version field

Parameter	Description
standard_version_t	Pointer to version structure to store current FLASH driver version information uint8_t bugfix; /* bugfix version [7:0] */ uint8_t minor; /* minor version [15:8] */ uint8_t major; /* major version [23:16] */ char name; /* character name is "F", where "F" stands for Flash and character version cannot be changed, it is fixed */

Example:

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define FLASH_API_TREE ((flash_driver_interface_t*)ROM_API_TREE[4])
uint32_t FlashDriverVersion = FLASH_API_TREE-> version;
```

In this SoC, the FLASH Driver version is "0x46010100", it means the FLASH driver version is 1.1.0.

13.2.2.2 flash_init

This API initializes the FLASH default parameters and related FLASH clock for the FLASH and FMC. The *flash_init* API should be called before all the other FLASH APIs.

Prototype :

```
status_t (*flash_init)(flash_config_t *config);
```

Table 121. Parameters used for flash_init API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example :

```
flash_config_t flashConfig;
uint32_t status = FLASH_API_TREE->flash_init (&flashConfig);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means flash has been inited successfully.

13.2.2.3 flash_erase

This API erases the internal FLASH region specified by the parameters. This API must be called after the flash_init.

Prototype :

```
status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t key);
```

Table 122. Parameters used in flash_erase API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be erased. The start address must be sector size aligned (that is, a multiple of 8 KB).
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be sector size aligned.
key	Key is used to validate erase operation. Must be set to “kFLASH_ApiEraseKey”. kFLASH_ApiEraseKey = (((‘k’) << 24) | ((‘e’) << 16) | ((‘f’) << 8) | ((‘l’)))

Example :

```
uint32_t start = 0x0 ;
uint32_t lengthInBytes = 0x2000 ;
#define ERASE_KEY 0x6b65666b
uint32_t status = FLASH_API_TREE->flash_erase (&flashConfig, start, lengthInBytes, ERASE_KEY);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the FLASH region 0x0 – 0x2000 has been erased.

13.2.2.4 flash_program

This API programs the data into the internal flash page specified by input parameters. The required "start" and "lengthInBytes" parameters must be page-size aligned. This API must be called after the flash_init. Flash must be erased before programming.

Prototype :

```
status_t (*flash_program)(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t
lengthInBytes);
```

Table 123. Parameters used in flash_program API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be programmed. The start address must be 128 bytes-aligned.
src	Pointer to the source buffer of data that is to be programmed into flash.
lengthInBytes	The length in bytes (not words or long words) to be programmed; the length must also be 128 bytes-aligned.

Example:

```
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
```

```
    programBuffer[i] = (uint8_t)(i & 0xFF);
}status = FLASH_API_TREE->flash_program(&flashConfig, 0x0, programBuffer, sizeof(programBuffer);)
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the specified data has been programmed into the FLASH region.

13.2.2.5 flash_verify_erase

This API checks whether the specified FLASH area is in the erasure state.

This API must be called after the flash_init.

Prototype :

```
status_t (*flash_verify_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes);
```

Table 124. Parameters used in flash_verify_erase API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be verified. Must be page-aligned.
lengthInBytes	The length, given in bytes (not words or long words) to be verified. Must be page-aligned.

Example:

```
uint32_t start = 0x0 ;
uint32_t lengthInBytes = 0x1000 ;
uint32_t status = FLASH_API_TREE->flash_verify_erase (&flashConfig, start, lengthInBytes);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the internal FLASH region 0x0 – 0x1000 is erased.

13.2.2.6 flash_verify_program

This API checks whether the data in the specified area is identical to the data supposed to be programmed. This API must be called after the flash_init.

Prototype :

```
status_t (*flash_verify_program)(flash_config_t *config,
uint32_t start,
uint32_t lengthInBytes,
const uint8_t *expectedData,
uint32_t *failedAddress,
uint32_t *failedData);
```

Table 125. Parameters used in flash_verify_program API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be verified.
lengthInBytes	The length, given in bytes (not words or long words) to be verified.
expectedData	Pointer to the expected data that is to be verified against.
failedAddress	Pointer to returned first failing address.
failedData	Pointer to return failing data.

Example:

```
uint32_t start = 0x0 ;
uint32_t lengthInBytes = sizeof(expected_buffer) ;
uint32_t failedAddress, failedData;
uint32_t status = FLASH_API_TREE->flash_verify_erase (&flashConfig, start, lengthInBytes, (const
uint8_t *) expected_buffer, & failedAddress, & failedData);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the expected data same as the programed data in the specific FLASH region.

13.2.2.7 flash_get_property

This API returns the requested FLASH property. This API must be called after the flash_init and ffr_init.

Prototype :

```
status_t (*flash_get_property)(flash_config_t *config, flash_property_tag_t whichProperty, uint32_t
*value);
```

Table 126. Parameters used in flash_get_property API

Parameter	whichProperty	Description
config		Pointer to flash_config_t data structure in memory to store driver runtime state.
whichProperty		Pointer to the which flash property need to get.
value		Pointer to the returned flash property value.
Property definition		
kFLASH_PropertyPflashTotalSize	0x01	Pflash total size property.
kFLASH_PropertyPflashBlockSize	0x02	Pflash Block size property.

Table continues on the next page...

Table 126. Parameters used in flash_get_property API (continued)

Parameter	whichProperty	Description
kFLASH_PropertyPflashBlockCount	0x03	Pflash Block Count size property.
kFLASH_PropertyPflashBlockBaseAddr	0x04	flash block base address.
kFLASH_PropertyPflashPageSize	0x30	Pflash page size property.
kFLASH_PropertyFfrTotalSize	0x41	FFR total size property.
kFLASH_PropertyFfrBlockBaseAddr	0x42	FFR block base address property.
kFLASH_PropertyFfrPageSize	0x43	FFR page size property.

Example:

```
flash_property_tag_t whichProperty = kFLASH_PropertyPflashTotalSize;
uint32_t value = 0;
uint32_t status = FLASH_API_TREE->flash_get_property(&flashConfig, whichProperty, &value);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the flash property get successfully, and will return the property value if needed.

13.2.2.8 ffr_init

This API is used for initializing the FFR controller and the *flash_ffr_config* context. It must be called before calling other FFR APIs.

NOTE

flash_init must be called before calling the ffr_init API. Before calling other flash_driver_interface_t APIs, especially flash_verify_erase and flash_verify_program, it is necessary to call the flash_init and ffr_init APIs.

Prototype :

```
status_t (*ffr_init)(flash_config_t *config);
```

Table 127. Parameters used in ffr_init API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
uint32_t status = FLASH_API_TREE-> ffr_init (&flashConfig);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the FFR parameters has been configured.

13.2.2.9 ffr_lock

This API is used to enable the firewall for all FLASH banks, including enabling FLASH protection for three IFR banks and disabling write access to the FLASHBENKENABLE register. ffr_lock unconditionally locks writing to the CFPA and CMPA flash areas. Subsequent writes are inhibited unless a power-on reset (POR) or brown-out detect (BOD) reset occurs.

Prototype :

```
status_t (*ffr_lock)(flash_config_t *config);
```

Table 128. Parameters used in ffr_lock

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
uint32_t status = FLASH_API_TREE-> ffr_lock (&flashConfig);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the FFR regions has been locked.

13.2.2.10 ffr_cust_factory_page_write

The API is used for writing the CMPA data into the CMPA region, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_cust_factory_page_write)(flash_config_t *config, uint8_t *page_data, bool seal_part);
```

Table 129. Parameters used in ffr_cust_factory_page_write API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
page_data	Pointer to value address that will be written to the destination address.
seal_part	If set as true or the page_data includes the non-zero CMAC data, the CMPA CMAC will be calculated and programed into the CMPA region.

Example:

```
uint32_t cmpa_buffer [ffr_page_size] = {0, 2, 3, 4};
uint32_t status = FLASH_API_TREE-> ffr_cust_factory_page_write (&flashConfig, (uint8_t *)cmpa_buffer,
false);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the cmpa_buffer data has been programed into the CMPA region.

13.2.2.11 ffr_get_uuid

The API is used for getting the UUID data of the device, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_get_uuid)(flash_config_t *config, uint8_t *uuid);
```

Table 130. Parameters used in ffr_get_uuid API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
uuid	Pointer to value address, the value is read back from the Bank0_IFR1 0x0110_0000 – 0x0110_0010.

Example:

```
uint32_t uuid_buffer [4];
uint32_t status = FLASH_API_TREE-> ffr_get_uuid(&flashConfig, (uint8_t *)uuid_buffer);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the uuid data has been got from the UUID of the device.

13.2.2.12 ffr_get_customer_data

This API is used to read data stored in CMPA, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_get_customer_data)(flash_config_t *config, uint8_t *pData, uint32_t offset, uint32_t
len);
```

Table 131. Parameters used in ffr_get_customer_data API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Point to the destination buffer that stores data read from CMPA.
offset	Point to the offset value based on the CMPA start address(0x0100_4000) of the device.
len	The length in bytes to be read back. The offset + len <= 512B.

Example:

```
uint32_t cmpa_buffer[4];
uint32_t offset = 0;
uint32_t status = FLASH_API_TREE->ffr_get_customer_data(&flashConfig, (uint8_t *)cmpa_buffer, offset,
sizeof(cmpa_buffer));
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the CMPA data has been got from the CMPA region and stored into the cmpa_buffer.

13.2.2.13 ffr_cust_keystore_write

The API is used for programing the customer key store data into the customer key store region (0x0100_4160 - 0x0100_418f), and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_cust_keystore_write)(flash_config_t *config, ffr_key_store_t *pKeyStore);
```

Table 132. . Parameters used in ffr_cust_keystore_write API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pKeyStore	Pointer to the customer key store data buffer, which will be programed into the customer key store region.

Example:

```
uint32_t status = FLASH_API_TREE->ffr_cust_keystore_write(&flashConfig, (ffr_key_store_t *)
cust_keystore_buffer);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means customer key store data has been programed into the customer key store region.

13.2.2.14 ffr_infield_page_write

The API is used for programing the CFPA data into the CFPA inactive page and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_infield_page_write)(flash_config_t *config, uint8_t *page_data, uint32_t valid_len);
```

Table 133. . Parameters used in ffr_infield_page_write API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
page_data	Pointer to the source buffer of data that is to be programed into the in-field page.
valid_len	The length in bytes to be programmed, the length must equal the page size.

Example:

```
uint32_t cfpa_buffer [128];
uint32_t status = FLASH_API_TREE-> ffr_infield_page_write (&flashConfig, (uint8_t *) cfpa_buffer,
sizeof(cfpa_buffer));
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the CFPA data has been programed into the CFPA Scratch region.

13.2.2.15 ffr_get_customer_infield_data

The API is used for getting the CFPA data from the FFR CFPA region, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_get_customer_infield_data)(flash_config_t *config, uint8_t *pData, uint32_t offset,
uint32_t len);
```

Table 134. Parameters used in ffr_get_customer_infield_data API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Point to the destination buffer of data that stores data read from customer In-field page.
offset	Pointer to the offset value based on the CFPA address (Ping page: 0x01000000, Pong page: 0x01002000) of the device.
len	Point to the length of data to read, and the offset + len <= 8192B.

Example:

```
uint32_t cfpa_buffer[4];
uint32_t offset = 0;
uint32_t status = FLASH_API_TREE->ffr_get_customer_infield_data (&flashConfig, (uint8_t *)
cfpa_buffer, offset, sizeof(cfpa_buffer ));
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the CFPA data has been got from the CFPA region and stored into the cfpa_buffer.

13.2.2.16 flash_read

The API is used for getting internal flash data, including the FLASH and FFR data, and the API should be called after the flash_init.

Prototype :

```
status_t (*flash_read)(flash_config_t *config, uint32_t start, uint8_t *dest, uint32_t lengthInBytes);
```

Table 135. Parameters used in flash_read API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	Point to the start address where data will be read.
dest	Pointer to the buffer used for storing the read data.
lengthInBytes	Point to the read data length

Example:

```
uint32_t start_addr = 0x1000;
uint8_t read_buffer[512] = {0};
uint32_t length = 512;
uint32_t status = FLASH_API_TREE-> flash_read(&flashConfig, start_addr, (uint8_t *) read_buffer,
length);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the expected read region has been loaded into the read_buffer.

13.2.2.17 flash_get_cust_keystore

The API is used for getting the customer key store data from the customer key store region (0x0100_4160 - 0x0100_418f), and the API should be called after the flash_init and ffr_init.

Prototype:

```
status_t (*flash_get_cust_keystore)(flash_config_t *config, uint8_t *pData, uint32_t offset, uint32_t
len);
```

Table 136. Parameters used in flash_get_cust_keystore API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Pointer to the customer key store data buffer, which got from the customer key store region.
offset	Point to the offset value based on the customer key store address (0x01004160)of the device.
len	Point to the length of the expected get customer key store data, and the offset + len <= 512B.

Example:

```
uint8_t cust_keystore_buffer[512] = {0};
uint32_t offset = 0;
uint32_t length = 512;
uint32_t status = FLASH_API_TREE-> flash_get_cust_keystore (&flashConfig, (uint8_t *)
cust_keystore_buffer, offset, length);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means the customer key store data has been got from the customer key store region and stored into the cust_keystore_buffer.

13.2.2.18 flash_deinit

This API deinitializes the FLASH default parameters and related FLASH clock for the FLASH and FMC. The flash_deinit API should be called after all the other FLASH APIs.

Prototype :

```
status_t (*flash_deinit)(flash_config_t *config);
```

Table 137. Parameters used in flash_deinit API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
flash_config_t flashConfig;
uint32_t status = FLASH_API_TREE->flash_deinit (&flashConfig);
```

See the possible status codes in Table 138. If the status is kStatus_FLASH_Success return value, it means FLASH has been deinited successfully.

13.2.2.19 Possible return codes of the FLASH APIs

Table 138. Return codes for FLASH APIs

Return code	Value	Description
kStatus_FLASH_Success	0	API is executed successfully
kStatus_FLASH_InvalidArgument	4	An invalid argument is provided
kStatus_FlashAlignmentError	101	Address or length does not meet the required alignment.
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FLASH_CommandFailure	105	The FMU_FSTAT[FAIL] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.
kStatus_FlashEraseKeyError	107	The key provided does not match the programmed flash key.
kStatus_FlashRegionExecuteOnly	108	The area of flash is protected as execute-only.
kStatus_FLASH_CommandNotSupported	111	Flash API is not supported
kStatus_FLASH_ReadOnlyProperty	112	The flash property is read-only
kStatus_FLASH_InvalidPropertyValue	113	The flash property value is out of range
kStatus_FLASH_EccError	116	A correctable or uncorrectable error during command execution
kStatus_FLASH_CompareError	117	Destination and source memory contents do not match
kStatus_FLASH_InvalidWaitStateCycles	119	The wait state cycle set to r/w mode is invalid

Table continues on the next page...

Table 138. Return codes for FLASH APIs (continued)

kStatus_FLASH_OutOfDateCfpaPage	132	CFPA page version is out of date
kStatus_FLASH_BlankIfrPageData	133	Blank page cannot be read
kStatus_FLASH_EncryptedRegionsEraseNotDoneAtOnce	134	Encrypted flash subregions are not erased at once
kStatus_FLASH_SealedFfrRegion	137	The FFR region is sealed
kStatus_FLASH_FfrRegionWriteBroken	138	The FFR Spec region is not allowed to be written discontinuously
kStatus_FLASH_NmpaAccessNotAllowed	139	The NMPA region is not allowed to be read/ written/erased
kStatus_FLASH_CmpaCfgDirectEraseNotAllowed	140	The CMPA Cfg region is not allowed to be erased directly
kStatus_FLASH_FfrBankIsLocked	141	The FFR bank region is locked
kStatus_FLASH_CfpaScratchPageInvalid	148	CFPA Scratch Page is invalid
kStatus_FLASH_CfpaVersionRollbackDisallowed	149	CFPA version rollback is not allowed

13.2.2.20 Flash APIs Example

The section provides examples to introduce the Flash APIs usage.

Example 1 (program the image data or specific data into internal flash region):

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define FLASH_API_TREE ((flash_driver_interface_t*)ROM_API_TREE[4])
flash_config_t flashConfig;
status_t status = kStatus_Success;
uint32_t load_address = 0x10000;
uint32_t erase_size = 0x800;
uint32_t s_buffer[512];
for (uint32_t i = 0; i < 512; i++)
{
    s_buffer[i] = i;
}
uint32_t FlashDriverVersion = FLASH_API_TREE->version;
debug_printf("Flash version= %x\r\n", FlashDriverVersion);
status = FLASH_API_TREE->flash_init(&flashConfig);
debug_printf("flash_init version= %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->ffr_init(&flashConfig);
debug_printf("ffr_init version= %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
```

```
status = FLASH_API_TREE->flash_erase(&FlashConfig, load_address, erase_size, kFLASH_ApiEraseKey);
debug_printf("flash_erase status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
/* call the flash_verify_erase api to check whether the flash region has been erased */
status = FLASH_API_TREE->flash_verify_erase(&FlashConfig, load_address, erase_size);
debug_printf("flash_verify_erase status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
/* here s_buffer data can also be replaced by the image data that is used for dual image boot */
status = FLASH_API_TREE->flash_program(&FlashConfig, load_address, (uint8_t *)s_buffer,
sizeof(s_buffer);
debug_printf("flash_program status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
/* call the flash_verify_program api to check whether the s_buffer data has been loaded into the
flash region */
uint32_t failedAddress, failedData;
status = FLASH_API_TREE->flash_verify_program (&FlashConfig, load_address, sizeof(s_buffer), (const
uint8_t *)s_buffer, &failedAddress, &failedData);
debug_printf("flash_verify_program status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
```

Example 2 (update the ffr data and get the data from the ffr region):

```


#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define FLASH_API_TREE ((flash_driver_interface_t*)ROM_API_TREE[4])
flash_config_t flashConfig;
uint32_t pflashBlockBase = 0;
uint32_t cmpa_key_store_address = 0x0100_4000;
uint8_t pData[512];
uint8_t pData1[512] ;
uint8_t uuid[16];
uint32_t ffr_buffer_len = 512 ;
status_t status = kStatus_Success;
status = FLASH_API_TREE->flash_init(&flashConfig);
debug_printf("flash_init version= %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->ffr_init(&flashConfig);
debug_printf("ffr_init version= %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->flash_get_property(&FlashConfig, kFLASH_PropertyPflashBlockBaseAddr,
&pflashBlockBase);
debug_printf("kFLASH_PropertyPflashBlockBaseAddr = %x\r\n", pflashBlockBase);
```

```
if (status != kStatus_FLASH_Success)
{
    return status;
}
/* call ffr_infield_page_write to load CfpaData into the CFPA inactive region; CfpaData is provided
by user; the version of CfpaData should be larger than or same as before */
status = FLASH_API_TREE->ffr_infield_page_write(&FlashConfig, (uint8_t *)&CfpaData, FFR_BUFFER_LEN);
debug_printf("ffr_infield_page_write status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->ffr_get_customer_infield_data(&FlashConfig, (uint8_t *)pData, 0,
ffr_buffer_len);
debug_printf("ffr_get_customer_infield_data status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
/* call ffr_cust_factory_page_write to load CmpaData into the CMPA region; CMpaData is provided by
user */
status = FLASH_API_TREE->ffr_cust_factory_page_write(&FlashConfig, (uint8_t *)CmpaData, false);
debug_printf("ffr_cust_factory_page_write status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->ffr_get_customer_data(&FlashConfig, (uint8_t *)pData, 0, ffr_buffer_len);
debug_printf("ffr_get_customer_data status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->ffr_get_uuid(&FlashConfig, uuid);
debug_printf("ffr_get_uuid status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
/* call ffr_keystore_write to load CMPA_KeyStore into the CMPA key_store region; CMPA_KeyStore is
provided by user */
status = FLASH_API_TREE->ffr_keystore_write(&FlashConfig, (ffr_key_store_t *)CMPA_KeyStore);
debug_printf("ffr_keystore_write status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->flash_get_cust_keystore(&FlashConfig, (uint8_t *)pData, 0, ffr_buffer_len);
debug_printf("flash_get_cust_keystore status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
status = FLASH_API_TREE->flash_read(&FlashConfig, cmpa_key_store_address, (uint8_t *)pData1,
ffr_buffer_len);
debug_printf("flash_read status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
```

```
status = memcmp((const void*) pData, (const void*) pData1, ffr_buffer_len);
debug_printf("cmpa key_store read data compare status = %x\r\n", status);
if (status != 0)
{
    return status;
}
/* call ffr_lock to lock the ffr region; the ffr region can not be accessed until the next reset;
call the flash_init  and the ffr_init APIs */
status = FLASH_API_TREE->ffr_lock(&FlashConfig);
debug_printf("ffr_lock status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
    return status;
}
```

13.2.3 runBootloader API

The ROM bootloader provides an API for the user application to enter the ISP mode based on the designated ISP interface mode.

Prototype

```
void (*runBootloader)(void *arg);
```

Table 139. runBootloader API arg fields

Field	Offset	Description	
Tag	[31:24]	Fixed value: 0xEB (Enter Boot)	
Boot Mode	[23:20]	0 - Master boot mode	
		1 - Enter ISP mode	
ISP Interface	[19:16]	**ISP boot mode**	**Master boot mode**
		0 - Auto detection	0 – Internal flash boot
		1 - UART	2 – FlexSPI flash boot
		2 - SPI	Reserved
		3 - I2C	Reserved
		4 - USB FS HID	
		5 - USB HS HID	
		6 - CAN	
Instance	[15:12]	FlexSPI port selection 0: PortA 1: PortB	
Image Index	[11:08]	Boot image index selection in the dual image mode	
Reserved	[07:00]		

Example:

In the application image boot process, regardless of whether the ISP pin is connected to the high level, the device will directly enter the ISP mode through the UART interface according to the parameter ISP Interface in arg.

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define RUN_BOOTLOADER_API_TREE ((void (*)(void *)) ROM_API_TREE[0])
uint32_t arg = 0xEB120000;   /*0xEB: represents Enter Boot; 0x12: represents enter ISP mode by UART
only */
RUN_BOOTLOADER_API_TREE->runBootloader(&arg);
```

After the application image, which calls the above runBootloader API, has booted successfully, the device will only allow the UART interface to be connected to transfer the data with the host.

13.2.4 FLEXSPI FLASH Driver APIs

The FLEXSPI NOR Driver API set consists of APIs that can simplify external FLASH support. The API set provides the following features:

- Recognize an external Serial NOR FLASH device based on the JESD216 or above revision
- Probe the Octal Serial NOR FLASH via simplified option
- Program data to or read data from an external Serial NOR FLASH device
- Partially or fully erase an external Serial NOR FLASH device.
- Support for most Serial NOR FLASH devices

13.2.4.1 General description

The FlexSPI FLASH APIs are organized in the flexspi_nor_flash_driver_t structure. See the details in data structure below. See the details of each API in the subsequent sections.

```
typedef struct
{
    uint32_t version;
    status_t (*init)(uint32_t instance, flexspi_nor_config_t *config);
    status_t (*page_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const
uint32_t *src);
    status_t (*erase_all)(uint32_t instance, flexspi_nor_config_t *config);
    status_t (*erase)(uint32_t instance, flexspi_nor_config_t *config, uint32_t start, uint32_t
length);
    status_t (*erase_sector)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
    status_t (*erase_block)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
    status_t (*get_config)(uint32_t instance, flexspi_nor_config_t *config,
serial_nor_config_option_t *option);
    status_t (*read)(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t start,
uint32_t bytes);
    status_t (*xfer)(uint32_t instance, flexspi_xfer_t *xfer);
    status_t (*update_lut)(uint32_t instance, uint32_t seqIndex, const uint32_t *lutBase, uint32_t
numberOfSeq);
    status_t (*set_clock_source)(uint32_t clockSrc);
    void (*config_clock)(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode);
    status_t (*partial_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr,
const uint32_t *src, uint32_t length);
} flexspi_nor_flash_driver_t;
```

Each FLEXSPI FLASH API depends on a common *flexspi_nor_config_t* parameter to perform the proper operation. The *flexspi_nor_config_t* data structure is defined as below. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.

```
/* Serial NOR configuration block */

typedef struct _flexspi_nor_config
{
    flexspi_mem_config_t memConfig; /* Common memory configuration info via FlexSPI */
    uint32_t pageSize;              /* Page size of Serial NOR */
    uint32_t sectorSize;            /* Sector size of Serial NOR */
    uint8_t ipcmdSerialClkFreq;     /* Clock frequency for IP command */
    uint8_t isUniformBlockSize;     /* Sector/Block size is the same */
    uint8_t isDataOrderSwapped;     /* Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2) */
    uint8_t reserved0[1];           /* Reserved for future use */
    uint8_t serialNorType;          /* Serial NOR FLASH type: 0/1/2/3 */
    uint8_t needExitNoCmdMode;      /* Need to exit NoCmd mode before other IP command */
    uint8_t halfClkForNonReadCmd;   /* Half the Serial Clock for non-read command: true/false */
    uint8_t needRestoreNoCmdMode;   /* Need to Restore NoCmd mode after IP commmand execution */
    uint32_t blockSize;             /* Block size */
    uint32_t flashStateCtx;         /* FLASH State Context */
    uint32_t reserved2[10];         /* Reserved for future use */
} flexspi_nor_config_t;
```

13.2.4.2 version

This field indicates the FlexSPI FLASH driver version number.

Prototype :

```
uint32_t version;
```

Example:

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define FLEXSPI_FLASH_API_TREE ((flexspi_nor_flash_driver_t *)ROM_API_TREE[11])
uint32_t FlexSPIFLASHDriverVersion = FLEXSPI_FLASH_API_TREE -> version;
```

In this device, the FlexSPI FLASH Driver version is “0x00010802”, it means the FLEXSPI FLASH API version is 1.8.2.

13.2.4.3 flexspi_nor_flash_init

The API is used for initializing the FlexSPI controller based on the parameters. It requires a full flexspi_nor_config_t data structure to initialize the FLEXSPI controller, clocks, FLASH etc. The data structure can be generated via flexspi_nor_get_config API or in other manners.

Prototype :

```
status_t (*init)(uint32_t instance, flexspi_nor_config_t *config);
```

Table 140. Parameters used in the flexspi_nor_flash_init API

Parameter	Description
instance	Point to the FlexSPI FLASH controller instance. Only instance 0 is supported on this device.
config	This parameter includes the first parameter flexspi_mem_config_t pointing to the flash configuration block(FCB) and related external FLASH device parameters, which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.

Example:

```
#define FLEXSPI_INSTANCE (0)
flexspi_nor_config_t flashConfig;
// get the valid flashConfig data here
// Call the API with proper flashConfig parameter
uint32_t status = FLEXSPI_FLASH_API_TREE->init (FLEXSPI_INSTANCE, &flashConfig);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means FlexSPI FLASH driver initialization is completed successfully.

13.2.4.4 flexspi_nor_flash_page_program

The API is used for programming the page data into the Serial NOR FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*page_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const
uint32_t *src);
```

Table 141. Parameters used in flexspi_nor_flash_page_program API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB), consisting of arguments related to an external FLASH device, such as flash size, page size, and sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
dstAddr	The start address of the required FLASH memory to be programmed.
src	Pointer to the source buffer of data that is to be programmed into FLASH. The src address must be 32-bit aligned

Example:

```
uint32_t dstAddr = 0x1000 ;
uint8_t programBuffer[256];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
    programBuffer[i] = (uint8_t)(i & 0xFF);
```

```
}
status = FLEXSPI_FLASH_API_TREE->page_program(FLEXSPI_INSTANCE, &flashConfig, dstAddr,
&programBuffer);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the data has been written into the specified region of the FLASH. It cannot guarantee the data has been successfully programmed into the FLASH device.

13.2.4.5 flexspi_nor_flash_erase_all

The API is used for erasing the entire external FLASH region, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase_all)(uint32_t instance, flexspi_nor_config_t *config);
```

Table 142. Parameters used in flexspi_nor_flash_erase_all API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.

Example:

```
status = FLEXSPI_FLASH_API_TREE-> erase_all (FLEXSPI_INSTANCE, &flashConfig);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means that the chip erase operation is completed on the FLASH side.

13.2.4.6 flexspi_nor_flash_erase

The API is used for erasing the specified region of the external FlexSPI FLASH region, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase)(uint32_t instance, flexspi_nor_config_t *config, uint32_t start, uint32_t length);
```

Table 143. Parameters used in flexspi_nor_flash_erase API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.

Table continues on the next page...

Table 143. Parameters used in flexspi_nor_flash_erase API (continued)

Parameter	Description
start	The start offset address of the required flash memory to be erased. The start address must be page-aligned (that is, a multiple of page of the specific FlexSPI Flash).
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be page-aligned.

Example:

```
uint32_t offset = 0x1000 ;
uint832_t lengthInBytes = 0x1000 ;
status = FLEXSPI_FLASH_API_TREE-> erase (FLEXSPI_INSTANCE, &flashConfig, offset, lengthInBytes);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the external FlexSPI FLASH region (0x80001000 – 0x80002000) has been erased successfully.

13.2.4.7 flexspi_nor_flash_erase_sector

The API is used for erasing the specified sector of the external FlexSPI FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase_sector)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
```

Table 144. Parameters used in flexspi_nor_flash_erase_sector

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
start	The parameter specifies the offset address of the sector to be erased.

Example:

```
uint32_t offset = 0x0 ;
status = FLEXSPI_FLASH_API_TREE-> erase_sector (FLEXSPI_INSTANCE, &flashConfig, offset);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the specified sector region of the external FlexSPI FLASH region has been erased successfully.

13.2.4.8 flexspi_nor_flash_erase_block

The API is used for erasing the specified block region of the external FlexSPI FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase_block)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
```

Table 145. Parameters used in flexspi_nor_flash_erase_block

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
start	The parameter specifies the offset address of the block to be erased.

Example:

```
uint32_t offset = 0x0 ;
status = FLEXSPI_FLASH_API_TREE-> erase_block (FLEXSPI_INSTANCE, &flashConfig, offset);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the specified block region of the external FlexSPI FLASH has been erased successfully.

13.2.4.9 flexspi_nor_get_config

The API is used for getting the FlexSPI NOR configuration block based on specified option. It can probe the FLASH type and generate the flexspi_nor_config_t for later full initialization. Typically, the flexspi_nor_flash_init will be performed with the flexspi_nor_config_t parameter probed by the flexspi_nor_get_config API.

Note:

This FlexSPI Flash API only support the Quad FlexSPI Flash type.

Prototype :

```
status_t (*get_config)(uint32_t instance, flexspi_nor_config_t *config, serial_nor_config_option_t
*option);
```

Table 146. Parameters used in flexspi_nor_get_config API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
option	This parameter provides a simplified option for the FlexSPI driver to probe the FLASH and get the FLASH parameters later. See the details of the flexspi_nor_config_t in FLEXSPI FLASH Driver APIs. Definitions of this block as below.

Table continues on the next page...

Parameter	Description

```
typedef struct _serial_nor_config_option
{
    union
    {
        struct
        {
            uint32_t max_freq : 4;              //!< Maximum supported Frequency
            uint32_t misc_mode : 4;            //!< miscellaneous mode
            uint32_t quad_mode_setting : 4;   //!< Quad mode setting
            uint32_t cmd_pads : 4;           //!< Command pads
            uint32_t query_pads : 4;        //!< SFDP read pads
            uint32_t device_type : 4;      //!< Device type
            uint32_t option_size : 4;     //!< Option size, in terms of uint32_t,size = (option_size
+ 1)* 4
            uint32_t tag : 4;            //!< Tag, must be 0x0E
        } B;
        uint32_t U;
    } option0;

    union
    {
        struct
        {
            uint32_t dummy_cycles : 8;    //!< Dummy cycles before read
            uint32_t status_override : 8; //!< Override status register value during device mode
configuration
            uint32_t pinmux_group : 4;    //!< The pinmux group selection
            uint32_t dqs_pinmux_group : 4; //!< The DQS Pinmux Group Selection
            uint32_t drive_strength : 4;   //!< The Drive Strength of FlexSPI Pads
            uint32_t flash_connection : 4; //!< FLASH connection option: 0 - Single FLASH connected
to port A, 1 - Reserved, 2 - Single FLASH connected to Port B
        } B;
        uint32_t U;
    } option1;

} serial_nor_config_option_t;
```

Table 147. serial_nor_config_option_t option

Offset	Field	Description
0	Option0	See option0 definition for more details
4	Option1	Optional, effective only if the option Size field in Option0 is non-zero See option1 definition for more details.

Table 148. option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C

Table continues on the next page...

Table 148. option0 definition (continued)

Field	Bits	Description
option_size	27:24	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 6 - Micron Octal DDR 8 - Adesto EcoXiP DDR
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0 - 1 2 - 4 3 – 8
cmd_pad	15:12	Data pads during FLASH access command 0 - 1 2 - 4 3 – 8
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configure 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 via 0x31 command **NOTE** This field will be effective only if the device is compliant with JESD216 only 5 - (9 longword SDFP table)
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance

Table continues on the next page...

Table 148. option0 definition (continued)

Field	Bits	Description
		3 - Data Order Swapped mode (for MXIC OctaFLASH only) 5 - Select the FlexSPI data sample source as internal loop back; see FlexSPI usage for more details 6 - Configure the FlexSPI NOR flash memory running at stand SPI mode **NOTE** Experimental feature only. Do not use in production: keep it as 0 for normal usage.
max_freq	3:0	Max FLASH Operation speed 0 - Don't change the FlexSPI clock setting

Table 149. option1 definition

Field	Bits	Description
flash_connection	31:28	FLASH connection option: • 0 - Single FLASH connected to port A • 1 - Reserved • 2 - Single FLASH connected to Port B
Reserved	27:24	**NOTE** This field does not have any effect in this device.
dqs_pinmux_group	23:20	The DQS Pinmux Group Selection
pinmux_group	19:16	The pinmux group selection
status_override	15:8	Override status register value during device mode configuration
dummy_cycles	7:0	Dummy cycles for a read command • 0 - Use detected dummy cycle • Others - dummy cycles provided in the flash datasheet

Typical Option configurations are as below:

- QUAD NOR - Quad SDR Read: option0 = 0xc0000004 (75MHz)
- QUAD NOR - Quad DDR Read: option0 = 0xc0100003 (60MHz)

Example:

```
#define FLASH_OPTION_QSPI_SDR 0xc0000001
serial_nor_config_option_t flashConfigOption = { .option0 = { .U = FLASH_OPTION_QSPI_SDR } };
status = FLEXSPI_FLASH_API_TREE-> get_config(FLEXSPI_INSTANCE, &flashConfig, &flashConfigOption);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the FlexSPI FLASH config block has been configured based on the provided option successfully.

13.2.4.10 flexspi_nor_flash_read

The API is used for getting the FlexSPI NOR Flash data and the API should be called only after the FLEXSPI IP has been properly configured.

Prototype :

```
status_t (*read)(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t start,
uint32_t bytes);
```

Table 150. Parameters used in flexspi_nor_flash_read API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
dst	Pointer to the buffer used for storing the read data. The dst address must be 32-bit aligned
start	The offset address of the required flash memory to be read.
lengthInBytes	The length, given in bytes (not words or long words) to be read. Must be page-aligned.

Example:

```
uint32_t start_offset = 0x1000;
uint32_t read_buffer[512] = {0};
status = FLEXSPI_FLASH_API_TREE->read(FLEXSPI_INSTANCE, &flashConfig, & read_buffer, start_offset,
sizeof(read_buffer));
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the FlexSPI FLASH (0x80001000 – 0x80001800) has been read and stored the data into the read_buffer successfully.

13.2.4.11 flexspi_command_xfer

The API is used for performing the FlexSPI command, this API should be called only when the FLEXSPI IP is properly configured.

Prototype :

```
typedef enum _FlexSPIOperationType
{
    // FlexSPI operation: Only command, both TX and RX buffer are ignored.
    kFlexSpiOperation_Command,
    // FlexSPI operation: Configure device mode, the TX FIFO size is fixed in LUT.
    kFlexSpiOperation_Config,
    // FlexSPI operation: Write,  only TX buffer is effective
    kFlexSpiOperation_Write,
    // FlexSPI operation: Read, only Rx Buffer is effective.
    kFlexSpiOperation_Read,
```

```
    kFlexSpiOperation_End,
} flexspi_operation_t;

//!@brief FlexSPI Transfer Context
typedef struct _FlexSpiXfer
{
    flexspi_operation_t operation; // FlexSPI operation
    uint32_t baseAddress;          // FlexSPI operation base address
    uint32_t seqId;                // Sequence Id
    uint32_t seqNum;               // Sequence Number

    bool Reserved;     // This field does not have any effect in the ROM boot flow.
    const uint32_t *txBuffer;      // Tx buffer
    uint32_t txSize;               // Tx size in bytes
    uint32_t *rxBuffer;            // Rx buffer
    uint32_t rxSize;               // Rx size in bytes
} flexspi_xfer_t;

status_t flexspi_command_xfer(uint32_t instance, flexspi_xfer_t *xfer);
```

Table 151. Parameters used in flexspi_command_xfer API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
xfer	This parameter pointer to FlexSPI Transfer Context which consists of arguments related to FlexSPI operation, sequence, TX and RX related parameters.

Example:

```
uint32_t start_offset = 0x1000;
uint32_t read_buffer[512] = {0};
flexspi_xfer_t flashXfer;
flashXfer.operation = kFlexSpiOperation_Read;
flashXfer.seqNum = 1;
flashXfer.seqId = NOR_CMD_LUT_SEQ_IDX_READ;
while (sizeof(read_buffer))
{
    uint32_t readLength = bytes > 65535 ? 65535 : bytes;
    flashXfer.baseAddress = start_offset;
    flashXfer.rxBuffer = read_buffer;
    flashXfer.rxSize = readLength;
    status = FLEXSPI_FLASH_API_TREE->flexspi_command_xfer (instance, &flashXfer);
    if (status != kStatus_Success)
    {
        return status;
    }
    bytes -= readLength;
    start_offset += readLength;
    read_buffer += readLength / sizeof(uint32_t);
}
return status;
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the FlexSPI FLASH command_xfer has executed the read command successfully, and the read data (0x80001000 – 0x80001800) has been load into the read_buffer array.

NOTE

This API relies on the valid LUT to perform the supported operations. If the LUT is invalid, the result is unpredictable.

13.2.4.12 flexspi_update_lut

The API is used for configuring the FlexSPI Lookup table before calling the flexspi_command_xfer API. This API should be called only when the FLEXSPI IP is properly configured.

Prototype :

```
status_t (*update_lut)(uint32_t instance, uint32_t seqIndex, const uint32_t *lutBase, uint32_t
numberOfSeq);
```

Table 152. Parameters used in flexspi_update_lut API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
seqIndex	This parameter pointer to FlexSPI Dedicated LUT Sequence Index for IP Command.
lutBase	This parameter pointer to FlexSPI Lookup table , which holds FLASH command sequences
numberOfSeq	This parameter points to FlexSPI Sequence Number

Example:

```
// Program Pattern
const uint32_t probe_pattern[4] = { 0x33221100, 0x77665544, 0xbbaa9988, 0xffeeddcc };
flexspi_xfer_t flashXfer;
flashXfer.baseAddress = sizeof(flexspi_nor_config_t);
flashXfer.seqId = NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM;
flashXfer.operation = kFlexSpiOperation_Write;
flashXfer.txBuffer = (uint32_t *)probe_pattern;
flashXfer.txSize = sizeof(probe_pattern);
flexspi_update_lut(instance,NOR_CMD_LUT_FOR_IP_CMD,&flashConfig-
>memConfig.lookupTable[4*flashXfer.seqId], flashXfer.seqNum);
flashXfer.seqId = NOR_CMD_LUT_FOR_IP_CMD;
status = flexspi_command_xfer(instance, &flashXfer);
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the flexspi_update_lut has execute successfully, and the probe_pattern data has been load into the (baseAddress + 0x80000000) region.

13.2.4.13 flexspi_set_clock_source

The API is used for configuring the FlexSPI clock source. This API should be called before calling the flexspi_not_get_config. This API should be called only when the FlexSPI IP is properly configured. This API only changes the FLEXSPI clock selector in the SYSCON register if the user needs to use it.

Prototype:

```
status_t (*set_clock_source)(uint32_t clockSrc);
```

Parameter	Description
clockSrc	Point to the FlexSPI select clock source Flexspi clock select: • 0000 - No clock • 0001 - PLL0 clock • 0010 - No clock • 0011 - FRO_HF • 0100 - No clock • 0101 - PLL1 clock • 0110 - USB PLL clock

Example:

```
uint32_t flexspi_clock_source = 0x0;
status = FLEXSPI_FLASH_API_TREE-> set_clock_source (flexspi_clock_source);
```

See the possible status codes in Table 194. If the status is kStatus_Success return value, it means the set_clock_source has execute successfully, and the FlexSPI has configure the clock source from the configured clock before.

13.2.4.14 flexspi_config_clock

The API is used for configuring the FlexSPI clock. This API should be called only when the FLEXSPI IP is properly configured. flexspi_set_clock_source only changes the clock selector in SYSCON, flexspi_clock_config can configure the clock frequency of the FLEXSPI root clock and clock source.

Prototype:

```
void (*config_clock)(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode);
```

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
freqOption	This parameter pointer to FlexSPIFlexSPI flash serial clock frequency: kFlexSpiSerialClk_30MHz (1U) kFlexSpiSerialClk_50MHz (2U) kFlexSpiSerialClk_60MHz (3U) kFlexSpiSerialClk_75MHz (4U) kFlexSpiSerialClk_100MHz (5U)

sampleClkMode	This parameter pointer to configure the FlexSPI clock configuration type.
	kFlexSpiClk_SDR (0U) //!< Clock configure for SDR mode
	kFlexSpiClk_DDR (1U) //!< Clock configure for DDR mode

Example:

```
uint32_t flexspi_freqOption = 0x1;
uint32_t flexspi_sampleClkMode = 0x0;
status = FLEXSPI_FLASH_API_TREE-> config_clock(FLEXSPI_INSTANCE, flexspi_freqOption,
flexspi_sampleClkMode);
```

See the possible status codes in Table 194. If the status is kStatus_Success return value, it means the config_clock has execute successfully, and the FlexSPI flash clock will be configured with the provided parameter.

13.2.4.15 flexspi_nor_flash_partial_program

The API is used for partially programming the data into the specified address of FlexSPI FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*partial_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const
uint32_t *src, uint32_t length);
```

Table 153. Parameters used in the flexspi_nor_flash_partial_program API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
dstAddr	The start address of the required FlexSPI flash memory to be programmed.
src	Pointer to the source buffer of data that is to be programmed into FlexSPI flash.
length	The program size of the source buffer.

Example:

```
uint32_t flashAhbAddr = 0x80001000UL;
uint32_t flashAhbBase = 0x80000000UL;
uint32_t flashOffset = flashAhbAddr - flashAhbBase;
uint32_t programBuffer[256 / sizeof(uint32_t)];
uint8_t *buf_u8 = (uint8_t*)programBuffer;
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
    *buf_u8++ = (uint8_t)(i & 0xFFU);
}
status = FLEXSPI_FLASH_API_TREE->partial_program(FLEXSPI_INSTANCE, &flashConfig, flashOffset,
&programBuffer, sizeof(programBuffer));
```

See the possible status codes in Table 154. If the status is kStatus_Success return value, it means the data has been written into the FlexSPI FLASH.

13.2.4.16 Possible return codes for FLEXSPI FLASH driver APIs

Table 154. Return codes for FLEXSPI FLASH APIs

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_Invalidargument	4	The requested command's argument is undefined.
kStatus_FLEXSPI_SequenceExecutionTimeout	6000	The FLEXSPI Sequence Execution timeout
kStatus_FLEXSPI_InvalidSequence	6001	The FLEXSPI LUT sequence invalid
kStatus_FLEXSPI_DeviceTimeout	6002	The FLEXSPI device timeout
kStatus_FLEXSPINOR_ProgramFail	20100	Page programming failure
kStatus_FLEXSPINOR_EraseSectorFail	20101	Sector Erase failure
kStatus_FLEXSPINOR_EraseAllFail	20102	Chip Erase failure
kStatus_FLEXSPINOR_WaitTimeout	20103	The execution time out
kStatus_FlexSPINOR_WriteAlignmentError	20105	Address alignment error
kStatus_FlexSPINOR_CommandFailure	20106	Erase/Program Verify Error
kStatus_FlexSPINOR_SFDP_NotFound	20107	SFDP table was not found call
kStatus_FLEXSPINOR_Unsupported_SFDP_Version	20108	Unrecognized SFDP version
kStatus_FLEXSPINOR_FLASH_NotFound	20109	Flash detection failure
kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	20110	DDR Read dummy probe failure

13.2.4.17 FlexSPI Flash APIs Example

The section will provide the example to introduce All the FlexSPI Flash APIs usage and help users understand and use more deeply. Example(Program the image or specific data into the FlexSPI Flash region):

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define FLEXSPI_FLASH_API_TREE ((flexspi_nor_flash_driver_t *)ROM_API_TREE[11])
#define FLEXSPI_INSTANCE 0
#define FLEXSPI_PROGRAM_OFFSET (0x1000u)
#define FLEXSPI_ERASE_SIZE (0x2000u)
#define MAX_PAGE_SIZE 512
flexspi_nor_config_t flashConfig;
```

```
uint32_t FlexSPIFLASHDriverVersion = FLEXSPI_FLASH_API_TREE->version;
debug_printf("FlexSPIFLASHDriverVersion version= %x\r\n", FlexSPIFLASHDriverVersion);

// choose the Quad SDR FlexSPI Flash type as the example

serial_nor_config_option_t flashConfigOption = { .option0 = { .U = 0xc0000001}};
status = FLEXSPI_FLASH_API_TREE->get_config(FLEXSPI_INSTANCE, &flashConfig, &flashConfigOption);
debug_printf("flexspi_nor_get_config API status=%d!\n",status);
if (status != kStatus_Success)
{
    return status;
}

//Init the FlexSPI Flash base on the flashConfig, which is configured from the get_config API
status = FLEXSPI_FLASH_API_TREE->init(FLEXSPI_INSTANCE, &flashConfig);
debug_printf("flexspi_nor_flash_init returned error:%d\n", status);
if (status != kStatus_Success)
{
    return status;
}

status = FLEXSPI_FLASH_API_TREE->erase(FLEXSPI_INSTANCE, &flashConfig, FLEXSPI_PROGRAM_OFFSET,
FLEXSPI_ERASE_SIZE)
debug_printf ("flexspi_nor_flash_erase returned status:%d\n", status);;
if (status != kStatus_Success)
{
    return status;
}

// Create the specific data programBuffer to program the data into the FlexSPI Flash, also the data
can be replaced by image data provided by user
uint32_t programBuffer[MAX_PAGE_SIZE / sizeof(uint32_t)];
uint8_t *pBuffer = (uint8_t *)&programBuffer[0];
for (uint32_t i = 0; i < sizeof(programBuffer); i++)
{
    *pBuffer++ = (uint8_t)(i & 0xFF);
}

status = FLEXSPI_FLASH_API_TREE->page_program(FLEXSPI_INSTANCE, &flashConfig, FLEXSPI_PROGRAM_OFFSET,
programBuffer);
debug_printf("flexspi_nor_flash_page_program returned error:%d\n", status);
if (status != kStatus_Success)
{
    return status;
}

uint32_t readBuffer[MAX_PAGE_SIZE];
status = FLEXSPI_FLASH_API_TREE->read(FLEXSPI_INSTANCE, &flashConfig, readBuffer,
FLEXSPI_PROGRAM_OFFSET,
flashConfig.pageSize);
debug_printf("flexspi_nor_flash_read status = %d\n", status);

if (status != kStatus_Success)
{
    return status;
}
else
{
    if (memcmp(readBuffer, programBuffer, flashConfig.pageSize) == 0)
    {
```

```
    debug_printf ("Verify programming by IP read: PASSED\n");
    }
}

status = FLEXSPI_FLASH_API_TREE->erase(FLEXSPI_INSTANCE, &flashConfig, FLEXSPI_PROGRAM_OFFSET,
FLEXSPI_ERASE_SIZE)
debug_printf ("flexspi_nor_flash_erase returned status:%d\n", status);;
if (status != kStatus_Success)
{
    return status;
}
```

13.2.5 OTP-eFuse APIs

13.2.5.1 Version

This fields indicates the version number of the OTP API in this device.

Prototype :

```
standard_version_t version;
```

Table 155. Definition of the version Field in OTP API tree

Parameter	Description
standard_version_t	Pointer to version structure to store current eFuse driver version information uint8_t bugfix; /* bugfix version [7:0] */ uint8_t minor; /* minor version [15:8] */ uint8_t major; /* major version [23:16] */ char name; /* character name is "E", where "E" stands for eFuse. Character name and version are fixed */

Example:

```
#define ROM_API_TREE ((uint32_t*)0x1303fc 00)
#define OTP_API_TREE ((efuse_driver_t *)ROM_API_TREE[12])
uint32_t otpDriverVersion = OTP_API_TREE-> version;
```

In this device, the OTP driver version is "0x45010100", it means the OTP driver version is 1.1.0.

Following is the efuse_driver_t definition:

```
/* @brief EFUSE driver API Interface */
typedef struct
{
    standard_version_t version; /* efuse driver API version number */
    status_t (*init)(void);
    status_t (*deinit)(void);
    status_t (*p_efuse_read)(uint32_t addr, uint32_t *data);
    status_t (*p_efuse_program)(uint32_t addr, uint32_t data);
```

```
} efuse_driver_t;
```

13.2.5.2 otp_init

The API is used for initializing the OTP clock. The *otp_init* API should be called before all the other OTP APIs.

Prototype :

```
status_t (*init)(void);
```

Example :

```
uint32_t status = OTP_API_TREE->init ();
```

See the possible status codes in Table 158. If the status is kStatus_Success return value, it means OTP driver initialization completed successfully.

13.2.5.3 otp_deinit

This API is used to disable OTP controller APB clock.

Prototype :

```
status_t (*deinit)(void);
```

Example :

```
uint32_t status = OTP_API_TREE->deinit ();
```

See the possible status codes in Table 158. If the status is kStatus_Success return value, it means OTP driver deinitialization completed successfully.

13.2.5.4 otp_read

This API is used to read data from OTP-eFuse controller.

Prototype :

```
status_t (*p_efuse_read)(uint32_t addr, uint32_t *data);
```

Table 156. Parameters used in otp_read API

Parameter	Description
addr	Pointer to the eFuse index that specifies the EFUSE word to be read out. The index 0 read data: 0 - 127 bits one word data (which 32bits word data in the 128bits will be changed to a byte) The index 1 read data: 128 - 255 bits one word data (which 32bits word data in the 128bits will be changed to a byte)

Table continues on the next page...

Table 156. Parameters used in otp_read API (continued)

Parameter	Description
	The index 2 read data: 256 - 383 bits one word data (which 32bits word data in the 128bits will be changed to a byte) The index 3 read data: 800 - 831 bits (one word data) The following index return data will follow the index 3, each word will increase the index by one.
data	Pointer to the buffer used for storing the read 32 bit data.

Example :

```
#define EFUSE_INDEX (0x0)
uint32_t eFuseData;
uint32_t status = OTP_API_TREE-> p_efuse_read (EFUSE_INDEX, & eFuseData);
```

See the possible status codes in Table 158. If the status is kStatus_Success return value, it means the OTP data read completed successfully.

13.2.5.5 otp_program

This API is used to program data to specified OTP Word.

Prototype :

```
status_t (*p_efuse_program)(uint32_t addr, uint32_t data);
```

Table 157. Parameters used in the otp_program API

Parameter	Description
addr	Pointer to the eFuse index that specifies the EFUSE word to be programmed
data	Pointer to the 32-bit source data used for programming into the eFuse region.

Example:

```
#define EFUSE_INDEX (43)  /* represents CUST Efuse Index */
uint32_t eFuseData = 0x01010101;
uint32_t status = OTP_API_TREE-> p_efuse_program (EFUSE_INDEX, & eFuseData);
```

See the possible status codes in Table 158. If the status is kStatus_Success return value, it means the data 0x01010101 has been programmed into the eFuse index 43 region.

13.2.5.6 Possible return codes of OTP API

Table 158. Return codes for OTP APIs

Error Code	Value	Description

Table continues on the next page...

Table 158. Return codes for OTP APIs (continued)

kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.

13.2.5.7 OTP-eFuse APIs Example

This section provides an example of OTP API usage.

NOTE

The OTP-eFuse program is only-program-once, user need to make sure the program data is valid value.

Example:

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define OTP_API_TREE ((efuse_driver_t *)ROM_API_TREE[12])
#define EFUSE_CUST_TEST_INDEX 43 /* The efuse read will return the efuse bit 2080 - 2111 bits data in
the efusemap file */

#define EFUSE_CMPA_CRC_INDEX 1
uint32_t otpDriverVersion = OTP_API_TREE->version;
debug_printf("otpDriverVersion version= %x\r\n", otpDriverVersion);
uint32_t status = OTP_API_TREE->init();
debug_printf("efuse init status= %x\r\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* The otp efuse value program should not program invalid data into the efuse region, otherwise the
device can boot fail and not behave as expected */

uint32_t fuseLCVal = 0;
status_t status = OTP_API_TREE->efuse_program(EFUSE_CUST_TEST_INDEX, fuseLCVal);
debug_printf("efuse_program LC state status= %x\r\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* Here crc value of the cmpa is just the example data, user needs to set the valid crc data of the
cmpa */

uint32_t fuseCMPACRCVal = 0x12345678;
status_t status = OTP_API_TREE->efuse_program(EFUSE_CMPA_CRC_INDEX, fuseCMPACRCVal);
debug_printf("efuse_program crc value of the CMPA status= %x\r\n", status);
if (status != kStatus_Success)
{
```

```
    return status;
}
uint32_t fuseLCVal1;
uint32_t fuseCMPACRCVal1;
status_t status = OTP_API_TREE->efuse_read(EFUSE_LC_STATE_INDEX, &fuseLCVal1);
debug_printf("efuse_read LC state status = %x \r\n", status);
if (status != kStatus_Success)
{
    return status;
}
status_t status = OTP_API_TREE->efuse_read(EFUSE_CMPA_CRC_INDEX, & fuseCMPACRCVal1);
debug_printf("efuse_program crc value of the CMPA status= %x \r\n", status,);
if (status != kStatus_Success)
{
    return status;
}
status_t status = OTP_API_TREE->deinit();
debug_printf("efuse deinit status= %x\r\n", status);
if (status != kStatus_Success)
{
    return status;
}
```

13.2.6 IAP APIs

13.2.6.1 General description

IAP is in–application–programming. The general IAP APIs call and execution flow is shown in Figure 55.

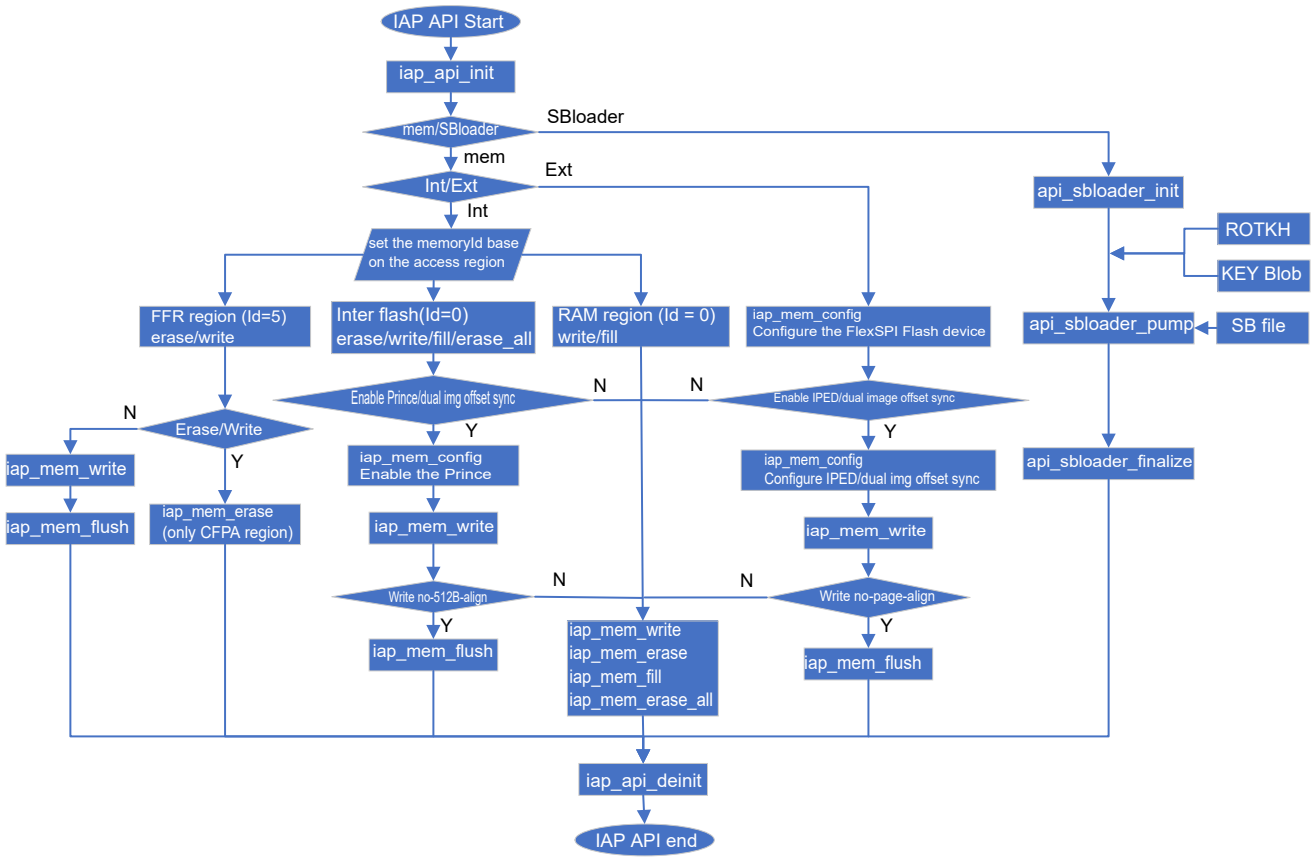


Figure 55. IAP API call flow

The bootloader IAP APIs are organized in the iap_api_interface_t structure, see the details from the below data structure. See the details of each API in the subsequent sections.

```
typedef struct iap_api_interface_struct
{
    standard_version_t version; /* IAP API version number */
    status_t (*api_init)(api_core_context_t *coreCtx, const kp_api_init_param_t *param);
    status_t (*api_deinit)(api_core_context_t *coreCtx);
    status_t (*mem_init)(api_core_context_t *ctx);
    status_t (*mem_read)(api_core_context_t *ctx, uint32_t addr, uint32_t len, uint8_t *buf, uint32_t
memoryId);
    status_t (*mem_write)(api_core_context_t *ctx, uint32_t addr, uint32_t len, const uint8_t *buf,
uint32_t memoryId);
    status_t (*mem_fill)(api_core_context_t *ctx, uint32_t addr, uint32_t len, uint32_t pattern,
uint32_t memoryId);
    status_t (*mem_flush)(api_core_context_t *ctx);
    status_t (*mem_erase)(api_core_context_t *ctx, uint32_t addr, uint32_t len, uint32_t memoryId);
    status_t (*mem_config)(api_core_context_t *ctx, uint32_t *buf, uint32_t memoryId);
    status_t (*mem_erase_all)(api_core_context_t *ctx, uint32_t memoryId);
    status_t (*sbloader_init)(api_core_context_t *ctx);
    status_t (*sbloader_pump)(api_core_context_t *ctx, uint8_t *data, uint32_t length);
    status_t (*sbloader_finalize)(api_core_context_t *ctx);
} iap_api_interface_t;
```

Each IAP API depends on the common context named api_core_context_t. The detail of the context is defined as below:

```
/* @brief The API context structure */

typedef struct api_core_context
{
uint32_t reserved0[10];
    uint32_t reserved1[3];
    flash_config_t flashConfig;
    flexspi_nor_config_t flexspinorCfg;
    uint32_t reserved2[2];
    uint32_t reserved3[1];
    nboot_context_t *nbootCtx;
    uint8_t *sharedBuf;
    uint32_t reserved4[8];

} api_core_context_t;
```

The kp_api_init_param_t is defined as below:

```
typedef struct kb_api_parameter_struct
{
    uint32_t allocStart;
    uint32_t allocSize;
} kp_api_init_param_t;
```

The nboot_context_t is defined as below:

```
typedef struct _nboot_context
{
    uint32_t buffer[0x2e8/sizeof(uint32_t)];
} nboot_context_t;
```

The IAP APIs require a dedicated RAM region as the buffer for each operation. The user application needs to prepare an unused RAM region organized as the *kp_api_init_param_t* structure, and pass this structure to the iap_api_init API. The allocSize in the iap_api_init API call is used to store the different flash context which is used in the api_core_context_t. The minimum size can be set to at least the size of (api_core_context_t).

These fields are managed by api_core_context_t parameter, the user application doesn't need to touch it.

NOTE

Before IAP APIs are used, caller needs to make sure that ELS clocks are enabled. This is required for correct operation of sbloader and iap_mem API on devices with security subsystem.

13.2.6.2 version

The version field indicates the IAP API version number in the device.

Prototype :

```
standard_version_t version;
```

Table 159. Definition of the version in IAP API tree

Parameter	Description

standard_version_t	Pointer to version structure to store current FLASH driver version information
	uint8_t bugfix; //!< bugfix version [7:0]
	uint8_t minor; //!< minor version [15:8]
	uint8_t major; //!< major version [23:16]
	char name; //!< character name is 0x4B = "K".

Example:

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13])
uint32_t IAPVersion = IAP_API_TREE-> version;
```

In this SoC, the version of IAP Driver is “0x4B030100”, it means the API version is 1.0.0.

13.2.6.3 iap_api_init

The API is used for initializing the IAP API context for the other IAP APIs, this API must be called prior to other IAP APIs.

Prototype :

```
status_t (*api_init)(api_core_context_t *coreCtx, const kp_api_init_param_t *param);
```

Table 160. Parameters used in iap_api_init API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
param	Pointer to IAP API initialization data structure, used for storing the data during execute the IAP APIs.

Example:

```
api_core_context_t apiCoreCtx;
const kp_api_init_param_t apiInitParam = { .allocStart = 0x30010000, .allocSize = 0x6000 };
uint32_t status = IAP_API_TREE->api_init(&apiCoreCtx, &apiInitParam);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the IAP API operating environment has been init successfully.

13.2.6.4 iap_api_deinit

The API is used for deinitializing the IAP API context.

Prototype :

```
status_t (*api_deinit)(api_core_context_t *coreCtx);
```

Table 161. Parameters used in iap_api_deinit API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

Example:

```
uint32_t status = IAP_API_TREE->api_deinit(&apiCoreCtx);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the IAP API operating environment has been freed and cannot call the other IAP APIs.

13.2.6.5 iap_mem_init

The API is used for initializing the memory interface, including the internal FLASH, RAM, FLEXSPI FLASH.

Prototype :

```
status_t (*iap_mem_init)(api_core_context_t *coreCtx);
```

Table 162. Parameters used in the iap_mem_init API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

Example:

```
uint32_t status = IAP_API_TREE->iap_mem_init(&apiCoreCtx);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the IAP API memory interfaces have been init successfully.

NOTE

The mem_init will also be called in the api_init API, so the mem_init no need to be called again after calling the api_init.

13.2.6.6 iap_mem_write

The API is used for programing the data into the internal FLASH, RAM or the external FlexSPI FLASH based on the parameters, such as start address and memoryId, and this API must be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_write(api_core_context_t *coreCtx, uint32_t start, uint32_t lengthInBytes, const
uint8_t *buf, uint32_t memoryId)
```

Table 163. Parameters used in iap_mem_write API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
start	Points to the program address.
lengthInBytes	Program data size in bytes. If the data size is less than 512 bytes or not an even multiple of 512 bytes, then the iap_mem_flush command must be called to write the remaining bytes.
buf	Pointer to source buffer data that will be programed.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FFR (0x5U) //!< FFR region ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

Example 1:

```
uint32_t start1 = 0x10000 ;
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
    programBuffer[i] = (uint8_t)(i & 0xFF);
}
uint32_t status = IAP_API_TREE->mem_write(&apiCoreCtx, start1, sizeof(programBuffer), (uint8_t *)
programBuffer, 0);
```

Example 2:

```
uint32_t start2 = 0x80001000 ;
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
    programBuffer[i] = (uint8_t)(i & 0xFF);
}
uint32_t status = IAP_API_TREE->mem_write(&apiCoreCtx, start2, sizeof(programBuffer), (uint8_t *)
programBuffer, 9);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the programBuffer data has been programed into the FLASH region in the example1 and programed into the external FlexSPI FLASH in the example2.

NOTE

- Before calling the mem_write API, the internal flash should be erased first by calling the iap_mem_erase.
- Before calling the mem_write API to program the data into the external FlexSPI FLASH, the external flash should be configured first by calling the iap_mem_config.
- If the programBufferis not 512 alignment or program the data into the ffr region, need to call the iap_mem_flush to program the last less than 512 data into the memory region.

NOTE

If the iap_mem_write used for the FlexSPI Flash data program, the program data should be padded page-alignment to use the iap_mem_flush API to load the expect program data.

13.2.6.7 iap_mem_fill

The API is used for filling the specified pattern data into the memory region with the specific data length. This API is mainly for filling the FLASH configuration option related words into the internal RAM. This API must be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_fill(api_core_context_t *coreCtx, uint32_t start, uint32_t lengthInBytes, uint32_t
pattern, uint32_t memoryId)
```

Table 164. Parameters used in iap_mem_fill API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
start	Points to the program address.
lengthInBytes	Points to the program data size. Date size must be a multiple of 4 because the pattern is 32-bit.
pattern	Points to 32-bit data pattern that will be programed.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FFR (0x5U) //!< FFR region ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

Example:

```
uint32_t start = 0x10000 ;
uint32_t pattern = 0xa5a55a5a;
uint32_t status = IAP_API_TREE->mem_fill(&apiCoreCtx, start, 0x2000, pattern, 0);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the pattern data has been programed into the FLASH region (0x10000 – 0x12000).

13.2.6.8 iap_mem_erase

The API is used for erasing the internal FLASH region, FLEXSPI FLASH region via the memory interface. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_erase(api_core_context_t *coreCtx, uint32_t start, uint32_t lengthInBytes, uint32_t
memoryId)
```

Table 165. Parameters used in iap_mem_erase API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
start	Points to the erase address.
lengthInBytes	Points to the erase size. The erase size must be 8KB alignment.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FFR (0x5U) //!< FFR region ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

Example:

```
uint32_t start = 0x10000 ;
uint32_t lengthInBytes = 0x2000;
uint32_t status = IAP_API_TREE->mem_erase(&apiCoreCtx, start, lengthInBytes, 0);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the FLASH region (0x10000 – 0x12000) has be erased.

13.2.6.9 iap_mem_erase_all

The API is used for erasing the entire internal FLASH or FLEXSPI FLASH based on the memoryId parameter. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_erase_all(api_core_context_t *coreCtx, uint32_t memoryId)
```

Table 166. Parameters used in iap_mem_erase_all API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

Example:

```
uint32_t status = IAP_API_TREE->mem_erase_all(&apiCoreCtx, 0);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the FLASH region has be erased all.

13.2.6.10 iap_mem_config

The API is used for configuring the PRINCE for internal FLASH, IPED for the FlexSPI FLASH; it is also used for configuring the FLexSPI FLASH and enabling the dual image start address synchronization feature. The SWAP flash feature updates the dual image without considering whether the current active image is running in the bank0 or bank1. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_config(api_core_context_t *coreCtx, uint32_t *config, uint32_t memoryId)
```

Table 167. Parameters used in iap_mem_config API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
config	Pointer to the configuration data which will be used for configuring the memory region with the specific encrypt feature based on the memory id.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

Example 1:

```
prince_prot_region_arg_t flashConfigOptionPrince = { .option = { .tag = 0x50 }, .start =
0x00010000, .length = 0x2000 };
uint32_t status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)&flashConfigOptionPrince, 0);
```

Example 2:

```
#define FLASH_OPTION_QSPI_SDR 0xc0000001
flexspi_iped_region_arg_t flashConfigOptionIped = { .option = { .tag = 0x49 }, .start =
0x80001000, .end = 0x80001800 };
uint32_t status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)& flashConfigOptionIped, 9);
serial_nor_config_option_t flashConfigOption = { .option0 = { .U = FLASH_OPTION_QSPI_SDR } };
status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)& flashConfigOption, 9);
```

Example 3:

```
typedef struct
{
    uint32_t reserved : 24;
    uint32_t tag : 8; // Fixed to 0x52 ('R')
} dual_image_update_option_t;
dual_image_update_option_t dualImgUpdateOption = { .tag = 0x52 };
uint32_t status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)& dualImgUpdateOption, 0);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means:

- The internal FLASH region 0x10000 – 0x12000 has been enable the PRINCE feature

- The external FlexSPI FLASH has been configured with the QSPI SDR mode and enable the region 0x80001000 – 0x80001800 IPED feature.
- The dual image start address synchronization feature has been enabled when using the dual image boot feature, the access address will be processed with the remap offset to access the actual address.

13.2.6.11 iap_mem_flush

The API is used after iap_mem_write for flushing the data stored in the memory buffer into the destination memory region. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_flush(api_core_context_t *coreCtx);
```

Table 168. Parameters used in iap_mem_flush API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

Example:

```
uint32_t cfpastart = 0x0100_0000 ;
uint8_t cfpaBuffer[512];
for (uint32_t i=0; i<sizeof(cfpaBuffer); i++)
{
    cfpaBuffer [i] = (uint8_t)(i & 0xFF);
}
uint32_t status = IAP_API_TREE->mem_write(&apiCoreCtx, cfpastart, sizeof(cfpaBuffer), (uint8_t *)
cfpaBuffer, 5);
if (status != kStatus_Success)
{
    return status;
}
uint32_t status = IAP_API_TREE->mem_flush(&apiCoreCtx);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the cfpaBuffer data has been programed into the CFPA region. In the process of the FFR program and when the data of programed into the internal and external FLASH region is not page-align, user need to call the iap_mem_flush after the iap_mem_write to store the data into the actual memory region.

13.2.6.12 api_sbloader_init

This API is used for initializing the sbloader state machine before calling the api_sbloader_pump. This API should be called after the iap_api_init API.

Prototype :

```
status_t api_sbloader_init(api_core_context_t *ctx);
```

Table 169. Parameters used in api_sbloader_init API

Parameter	Description
ctx	Pointer to IAP API core context structure.

Example:

```
uint32_t status = IAP_API_TREE-> sbloader_init (&apiCoreCtx);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the sbloader runtime environment has been initialized successfully.

13.2.6.13 api_sbloader_pump

This API is used for handling the secure binary (SB3.1 format) data stream, which is used for image update, lifecycle advancing, etc. This API should be called after the iap_api_init and api_sbloader_init APIs.

Prototype :

```
status_t api_sbloader_pump(api_core_context_t *ctx, uint8_t *data, uint32_t length);
```

Table 170. Parameters used in api_sbloader_pump API

Parameter	Description
ctx	Pointer to IAP API core context structure.
data	Pointer to source data that is the sb file buffer data.
length	Points to the size of the process buffer data.

NOTE

If the api_sbloader_pump is used for the FlexSPI Flash data program, the program data should be padded to page-alignment and add additional few data (one Byte) to load the expect program data successfully.

If the api_sbloader_pump is used for internal flash operation, both the flash prefetch buffer and cache should be disabled before calling this API.

Example:

```
/* load the SB file chunk-by-chunk into chunk_processing_buffer */

get_chunk_from_SB_file(uint8_t * chunk_processing_buffer, uint8_t * SBfile_incoming_chunk, uint32_t
chunk_size);
status = IAP_API_TREE-> sbloader_pump(context, & chunk_processing_buffer, chunk_size);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the sb data has been processed successfully.

13.2.6.14 api_sbloader_finalize

The API is used for finalizing the sbloader operations.

Prototype :

```
status_t api_sbloader_finalize(api_core_context_t *ctx)
```

Table 171. Parameters used in api_sbloader_finalize API

Parameter	Description
ctx	Pointer to IAP API core context structure.

Example:

```
status = IAP_API_TREE-> sbloader_finalize (context);
```

See the possible status codes in Table 172. If the status is kStatus_Success return value, it means the last sb data has programed successfully.

13.2.6.15 Possible return codes for IAP driver APIs

Table 172. Return and Error codes for IAP APIs

Error Code	Value	Description
kStatus_IAP_Success	0	IAP API execution succeeded
kStatus_IAP_Fail	1	IAP API execution failed
kStatus_IAP_InvalidArgument	100001	Invalid argument detected during API execution
kStatus_IAP_OutOfMemory	100002	The Heap size is not enough during API execution
kStatus_IAP_ReadDisallowed	100003	The read memory operation is disallowed during API execution
kStatus_IAP_CumulativeWrite	100004	The FLASH region to be programmed is not empty
kStatus_IAP_EraseFailure	100005	Erase operation failed
kStatus_IAP_CommandNotSupported	100006	The specific command is not supported
kStatus_IAP_MemoryAccessDisabled	100007	Memory access is disabled, typically occurred on the FLEXSPI NOR if it is not configured properly
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_Invalidargument	4	The requested command's argument is undefined.
kStatusRomLdrSectionOverrun	10100	means reached the end of the sb file processing
kStatusRomLdrSignature	10101	the signature or version are incorrect

Table continues on the next page...

Table 172. Return and Error codes for IAP APIs (continued)

Error Code	Value	Description
kStatusRomLdrSectionLength	10102	the bootOffset/ new section count is out of range
kStatusRomLdrEOFReached	10104	the end of the image file is reached
kStatusRomLdrChecksum	10105	The checksum of a command tag block is invalid.
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.
kStatusRomLdrUnknownCommand	10107	An unknown command was found in the SB file.
kStatusRomLdrIdNotFound	10108	There was no bootable section found in the SB file.
kStatusRomLdrDataUnderrun	10109	The SB state machine is waiting for more data.
kStatusRomLdrJumpReturned	10110	The function that was jumped to by the SB file has returned.
kStatusRomLdrCallFailed	10111	The call command in the SB file failed.
kStatusRomLdrKeyNotFound	10112	A matching key was not found in the SB file's key dictionary to unencrypt the section.
kStatusRomLdrSecureOnly	10113	The SB file sent is unencrypted, and security on the target is enabled.
kStatusRomLdrResetReturned	10114	The SB file reset operation has unexpectedly returned.
kStatusRomLdrRollbackBlocked	10115	An Image version rollback event detected
kStatusRomLdrInvalidSectionMacCount	10116	Invalid Section MAC count detected in the SB file
kStatusRomLdrUnexpectedCommand	10117	The command tag in the sb-file is unexpected
kStatusRomLdrBadSBKEK	10118	Bad SBKEK detected

13.2.6.16 IAP API examples

The section will provide the example to introduce all the IAP APIs usage and help users understand and use more deeply.

Example 1 (program the image or specific data into internal Prince enable region and update ffr region):

```
#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13])
#define FLASH_BUFFER_LEN 0x500
#define CFPA_SCRATCH_ADDRESS 0x3dc00U
#define CMPA_ADDRESS 0x3e200U
#define CMPA_KEY_STORE_ADDRESS 0x3e400U
#define FLASH_PROGRAM_ADDRESS 0x10000;
#define FLASH_ERASE_LEN 0x2000;
#define flashMemId 0
#define ffrMemId 5

uint32_t s_buffer[FLASH_BUFFER_LEN];
```

```
for (uint32_t i = 0; i < FLASH_BUFFER_LEN; i++)
{
    s_buffer[i] = i;
}

uint32_t IAPVersion = IAP_API_TREE->version;
debug_printf("IAPVersion = %x\r\n", IAPVersion);
uint32_t status = kStatus_Success;
api_core_context_t apiCoreCtx;

/* User needs to provide the apiInitParam, which is used as the RAM region for the IAP APIs call */

const kp_api_init_param_t apiInitParam = { .allocStart = 0x30010000, .allocSize = 0x6000 };
status = IAP_API_TREE->api_init(&apiCoreCtx, &apiInitParam);

debug_printf("IAP API api_init return status=%d!\n",status);
if (status != kStatus_Success)
{
    return status;
}

/* config the internal flash region 0x10000 - 0x12000 with the prince feature to encrypt the data in
the region */

prince_prot_region_arg_t flashConfigOptionPrince = { .option = { .tag = 0x50 }, .start =
FLASH_PROGRAM_ADDRESS, .length = FLASH_ERASE_LEN };
status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)&flashConfigOptionPrince, flashMemId);
debug_printf("iap_mem_config prince status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

status = IAP_API_TREE->mem_erase(&apiCoreCtx, FLASH_PROGRAM_ADDRESS, FLASH_ERASE_LEN, flashMemId);

debug_printf("iap_mem_erase return status=%d!\n",status);
if (status != kStatus_Success)
{
    return status;
}
status = IAP_API_TREE->mem_write(&apiCoreCtx, FLASH_PROGRAM_ADDRESS, sizeof(s_buffer), (uint8_t
*)s_buffer, flashMemId);
debug_printf("iap_mem_write status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* If load data is not 512B-aligned, user needs to call mem_flush to program the last remaining data
that is not 512B-aligned */

status = IAP_API_TREE->mem_flush(&apiCoreCtx);
debug_printf("iap_mem_flush status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* Here CfpaData is provided by the user and the version should be larger than the current cfpa
version */
```

```
status = IAP_API_TREE->mem_write(&apiCoreCtx, CFPA_SCRATCH_ADDRESS, sizeof(CfpaData), (uint8_t
*)&CfpaData, ffrMemId);
debug_printf("iap_mem_write cfpa data return status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* User should always call mem_flush after mem_write to program the ffr data */

status = IAP_API_TREE->mem_flush(&apiCoreCtx);
debug_printf("iap_mem_flush cfpa data status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* Here the CmpaData is provided by the user */

status = iap_mem_write(&apiCoreCtx, CMPA_ADDRESS, sizeof(CmpaData), (uint8_t *)&CmpaData, ffrMemId);
debug_printf("iap_mem_write status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* User should always call mem_flush after mem_write to program the ffr data */

status = IAP_API_TREE->mem_flush(&apiCoreCtx);
debug_printf("iap_mem_flush cfpa data status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

status = iap_api_deinit(&apiCoreCtx);
debug_printf("iap_api_deinit status = %x!\n", status);
if (status != kStatus_Success)
{
    return status;
}
```

Example 2 (FlexSPI Flash with IPED):

```

#define ROM_API_TREE ((uint32_t*)0x1303fc00)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13])
#define FLASH_OPTION_SEL 0xc0000001     /* Choose the QSPI SDR FlexSPI Flash option */
#define FLEXSPI_BASE_ADDR (0x80000000u) /* Flash Base Addresss for AHB access */
#define FLEXSPI_START_OFFSET (0x1000u)  /* Test address for IP access */
#define FLEXSPI_START_ADDR (FLEXSPI_BASE_ADDR + FLEXSPI_START_OFFSET) /* Test addresss for AHB access
*/
#define FLEXSPI_ERASE_SIZE (0x2000u)   /* Erase size in the demo */

uint32_t status = kStatus_Success;

api_core_context_t apiCoreCtx;

/* User needs to provide apiInitParam, which is used as the RAM region for the IAP APIs call */
```

```
const kp_api_init_param_t apiInitParam = { .allocStart = 0x30010000, .allocSize = 0x6000 };
status = IAP_API_TREE->api_init(&apiCoreCtx, &apiInitParam);

debug_printf("IAP API api_init return status=%d!\n",status);
if (status != kStatus_Success)
{
    return status;
}

/* Configure the FlexSPI flash region 0x80001000 - 0x80001800 with IPED feature */

uint32_t flexspiMemId = 9;
flexspi_iped_region_arg_t flashConfigOptionIped = { .option = { .tag = 0x49 }, .start =
0x80001000, .end = 0x80001800 };
status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)&flashConfigOptionIped, flexspiMemId);
debug_printf("iap_mem_config iped status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* Configure the FlexSPI flash device with the selected option */

serial_nor_config_option_t flashConfigOption = { .option0 = { .U = FLASH_OPTION_SEL}};
status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)&flashConfigOption, flexspiMemId);
debug_printf("iap_mem_config status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

status = IAP_API_TREE->mem_erase(&apiCoreCtx, FLEXSPI_START_ADDR, FLEXSPI_ERASE_SIZE, flexspiMemId);
debug_printf("iap_mem_erase returned status:%d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* Here the programBuffer data can also be replaced by the image data for app to boot in the FlexSPI
Flash boot mode */
uint32_t programBuffer[FLEXSPI_ERASE_SIZE / sizeof(uint32_t)];
uint8_t *pBuffer = (uint8_t *)&programBuffer[0];
for (uint32_t i = 0; i < sizeof(programBuffer); i++)
{
    *pBuffer++ = (uint8_t)(i & 0xFF);
}
status = IAP_API_TREE->mem_write(&apiCoreCtx, FLEXSPI_START_ADDR, sizeof(programBuffer),
(const uint8_t *)programBuffer, flexspiMemId);
debug_printf("iap_mem_write returned status:%d\n", status);
if (status != kStatus_Success)
{
    return status;
}
status = iap_api_deinit(&apiCoreCtx);
debug_printf("iap_api_deinit status = %x!\n", status);
if (status != kStatus_Success)
{
```

```
    return status;
}
```

Example 3 (update the dual-image into the internal flash):

```

#define ROM_API_TREE ((uint32_t*)0x1303e000)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13])
#define FLASH_PROGRAM_ADDRESS 0x0;
#define FLASH_ERASE_LEN 0x2000;
#define flashMemId 0

typedef struct
{
    uint32_t reserved : 24;
    uint32_t tag : 8; // Fixed to 0x52 ('R')
} dual_image_update_option_t;

uint32_t status = kStatus_Success;

api_core_context_t apiCoreCtx;

/* User needs to provide apiInitParam, which is used as the RAM region for the IAP APIs call */

const kp_api_init_param_t apiInitParam = { .allocStart = 0x30010000, .allocSize = 0x6000 };
status = IAP_API_TREE->api_init(&apiCoreCtx, &apiInitParam);

debug_printf("IAP API api_init return status=%d!\n",status);
if (status != kStatus_Success)
{
    return status;
}

/* Configure the dual-image update into specific lower version image region */

dual_image_update_option_t dualImgUpdateConfigOption = { .tag = 0x52 };
uint32_t flashMemId = 0x0U;
status = iap_mem_config(&apiCoreCtx, (uint32_t *)&dualImgUpdateConfigOption, flashMemId);
debug_printf("iap_mem_config status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* The erase address will be processed with the remap offset to access the actual address; if the
lower version image is stored in the 0x0, the erase address will be 0; if the lower version image is
stored in the remap offset region, the erase address will be 0x0 + remap_offset; User does not need
to set the erase address, which is just processed by API. */


status = IAP_API_TREE->mem_erase(&apiCoreCtx, FLASH_PROGRAM_ADDRESS, FLASH_ERASE_LEN, flashMemId);

debug_printf("iap_mem_erase return status=%d!\n",status);
if (status != kStatus_Success)
{
    return status;
}

/* The write address will be processed with the remap offset to access the actual address; if the
lower version image is stored in the 0x0, the write address will be 0; if the lower version image is
```

```
stored in the remap offset region, the erase address will be 0x0 + remap_offset. User does not need
to set the write address, which is just processed by API. */

status = IAP_API_TREE->mem_write(&apiCoreCtx, FLASH_PROGRAM_ADDRESS, sizeof(s_buffer), (uint8_t
*)s_buffer, flashMemId);
debug_printf("iap_mem_write status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}

/* If the load data is not 512B-aligned, user needs to call mem_flush to program the last remaining
data that is not 512B-aligned */

status = IAP_API_TREE->mem_flush(&apiCoreCtx);
debug_printf("iap_mem_flush status = %d\n", status);
if (status != kStatus_Success)
{
    return status;
}
```

13.2.6.17 SBloader APIs

This section uses the SBloader APIs demo to introduce the SBloader APIs usage and helps the user to know how to use the APIs. In addition, the section uses the practical application scenarios to introduce the convenience, flexibility, and practicality of the SBloader APIs. The flow diagram of the SBloader APIs call demo is as follow:

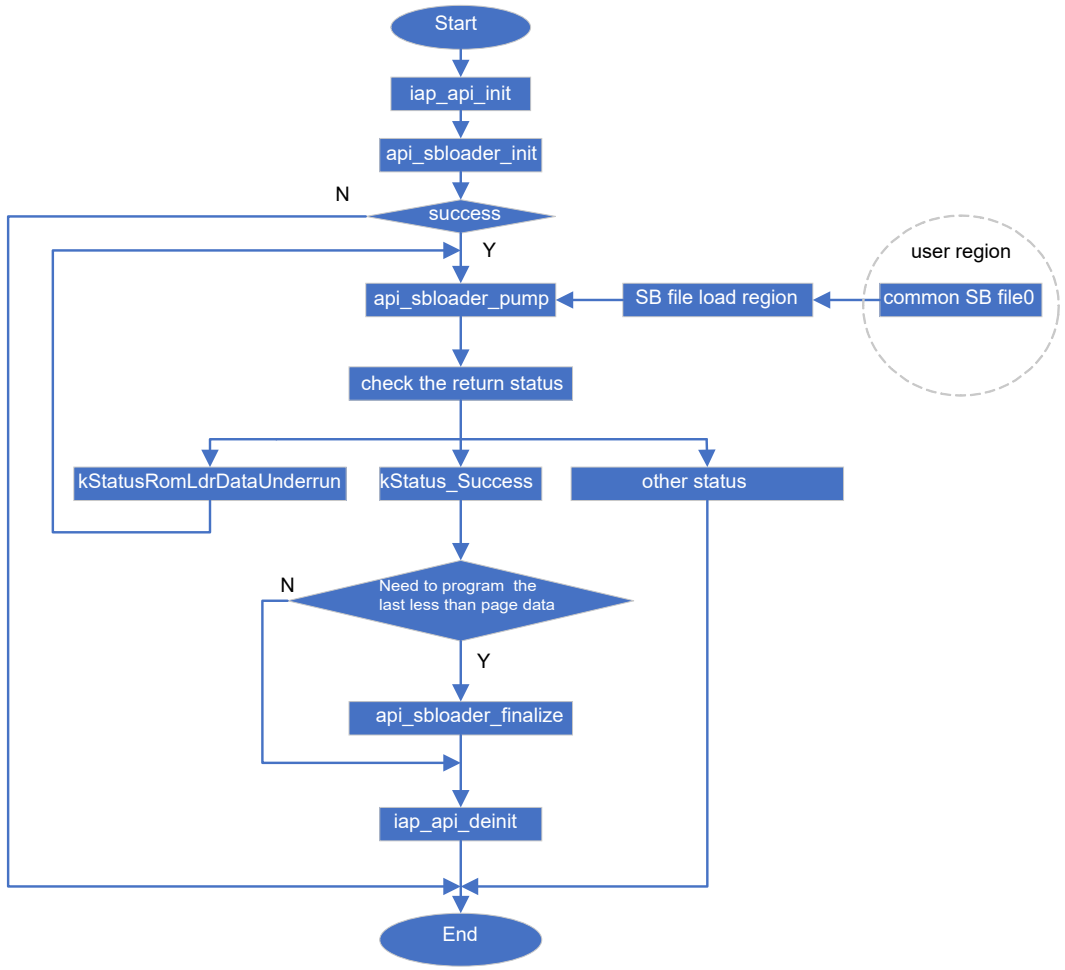


Figure 56. Flow diagram of the SBloader APIs call demo

The above flow diagram mainly introduces the SBloader APIs call flow. When the valid SB3.1 file has been generated and the necessary running environment (ROKTH and KeyBlob) has been setup up, the details can be found in Secure Boot ROM chapter. From the flow diagram, the iap_api_init will be called first to initialize and configure the IAP API core context. And then call the api_sbloader_init to initialize the loader state machine. If the api_sbloader_init return status is success, then the api_sbloader_pump will be called to process the SB file data stream, which is transferred from the user region (internal flash region, external flash region and RAM region apart from the reserved RAM region, list in the following).

1. 0x20008000 - 0x2000ffff
2. 0x30008000 - 0x3000ffff
3. 0x20010000 - 0x20017fff
4. 0x30010000 - 0x30017fff
5. 0x20060000 - 0x20067fff
6. 0x30060000 - 0x30067fff

In the running process of the api_sbloader_pump function, the return status includes three kinds of the results, which can be referred to in Table 172.Return and Error codes for IAP APIs. If the return status is kStatusRomLdrDataUnderrun, more data in the common SB_file0 should be transferred to the api_sbloader_pump to process, and the process has not reached the end of the SB file data. When the kStatus_Success has been received, common SB_file0 has been completely received and processed so

as to update the firmware in the SB_file0 successfully. The other status indicates that api_sbloader_pump API failed to execute. The iap_api_deinit will be executed to clear the IAP APIs context parameters.

In addition, in the process of executing the SB file in the api_sbloader_pump, only the load address of the SB file data will be transferred to the api_sbloader_pump, the data of the SB file is executed with the packet chunk. Users should get and load the packet chunk of the SB file into the ram region like the following function get_data_from_sbfile(), and then call the api_sbloader_pump to process the ram data repeatedly until to the end of the SB file. In transferring the SB file chunk to the device, the user can use different channels to receive the chunk data like UART, SPI, I2C, or USB interface.

Example:

```
#define ROM_API_TREE ((uint32_t*)0x1303e000)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13])
api_core_context_t apiCoreCtx = { .flashConfig.modeConfig.sysFreqInMHz = 96 };
const kp_api_init_param_t apiInitParam = { .allocStart = 0x30010000, .allocSize = 0x6000 };
uint32_t status = IAP_API_TREE->api_init(&apiCoreCtx, &apiInitParam);
if (status != kStatus_Success)
{
    return status;
}
uint32_t status = IAP_API_TREE->sbloader_init(&apiCoreCtx);
if (status != kStatus_Success)
{
    return status;
}
uint8_t user_buf[1024];
uint32_t size;
while ((status = get_data_from_sbfile(&buffer, &size)) == kStatus_Success)
{
    status = IAP_API_TREE->sbloader_pump(context, &buffer, size);
    if (status == kStatusRomLdrDataUnderrun)
    {
        /* continue to receive the data from SB file for processing */
        debug_printf("Warning: Success to execute ROM API erase, but need more data \n");
    }
    else if (status == kStatus_Success)
    {
        debug_printf("Completed: Success to execute ROM API erase\n");
        break;   /* finish the SB file firewarre update and break */
    }
    else if (status != kStatus_Success)
    {
        debug_printf("Error: Failed to run API kb_execute, status is hex(%x) \n", (uint32_t)status);
        return status;   /* return the fail status or other actions */
    }
}
status = IAP_API_TREE->sbloader_finalize (context);
if (status != kStatus_Success)
{
    debug_printf("Error: Failed to sbloader_finalize\n");
    return status;
}
```

Chapter 14
In-System Programming (ISP)

14.1 Overview

This chip includes ISP functions to support image programming from the serial interface (UART, I^2C, SPI, CAN) and USB HID.

NOTE

This chapter is also available in the *MCX N Security Reference Manual*, with differences related to security functionality. The version of this chapter in the reference manual does not include the security-relevant information.

14.1.1 Features

- Peripheral interfaces:
  - UART
  - I^2C
  - SPI
  - CAN
  - USB
- Automatic detection of the active peripheral
- UART peripheral implements auto-baud detection
- CAN peripheral implements auto-baud detection for predefined baud rates:
  - 1 Mbit/s
  - 500 kbit/s
  - 250 kbit/s
  - 125 kbit/s
- A common packet-based protocol for all peripherals
- Packet error detection and retransmission
- Flash-resident configuration options (in CMPA)
- RAM protection used by the bootloader while it is running
- Retrieval of the device properties, such as flash memory and RAM size
- Multiple options for executing the bootloader, either at system startup or under application control at runtime
- Support for internal flash memory access
- Support for encrypted image downloading
- External flash memory access
- In CMPA, user can use blhost tool command 'write-memory' to program or update the CMPA

14.2 Functional description

14.2.1 Bootloader

ISP stores the boot code in internal ROM. After a reset, the Arm processor starts its code execution from this memory. ISP executes the bootloader code every time the part is powered on, reset, or woken up from a deep power-down, low-power mode.

The bootloader provides a flash memory and OTP-eFuse programming utility that operates over a serial connection on the MCUs. It enables quick and easy programming of MCUs through the entire product lifecycle, including application development, final product manufacturing, and beyond. Host-side command line and GUI tools are available to communicate with the bootloader. Users can use host tools to upload and download application code and to perform manufacturing via the bootloader.

For information on bootloader operation and boot pin, see Overview.

14.2.2 In-system programming (ISP)

Serial booting and other related functions are supported in several different ways:

- ISP protocol
- Bootloader packet types
- Bootloader command set
- UART ISP
- I^2C ISP
- SPI ISP
- USB ISP
- CAN ISP

14.2.3 Memory map after reset

The boot ROM is located in the memory region beginning with the address 1300_0000h. Both the ISP and IAP software use parts of the on-chip RAM. Use of the RAM is described in ISP interrupt and SRAM use.

Based on the DEFAULT_ISP_MODE settings or ISP pin settings, the ROM enters ISP mode and can auto-detect activity on these interfaces:

- CAN
- I^2C
- SPI
- UART
- USB HID

The auto-detect looks for activity on these interfaces and selects the appropriate interface when a properly formed frame is received. If an invalid frame is received, the data is discarded and scanning resumes. Communications for these interfaces are described in In-System programming protocol and Bootloader packet types.

14.2.4 ISP interrupt and SRAM use

14.2.4.1 Interrupts during IAP

When the user application code starts executing, the interrupt vectors from the SRAM are active. Before making an IAP call, you must disable the interrupts. The IAP code does not use or disable interrupts.

14.2.4.2 RAM used by the ISP command handler

ISP reserves the regions designated below for bootloader use when the bootloader is running.

Table 173. RAM used by ROM during ROM execution

RAM region	Purpose
2000_8000h–2001_7FFFh 2001_8000h–2001_9FFFh	• BSS • RW • Stack • Heap 2001_8000h–2001_9FFFh (For the recovery sb loader file handling only)

14.2.5 ISP boot flow

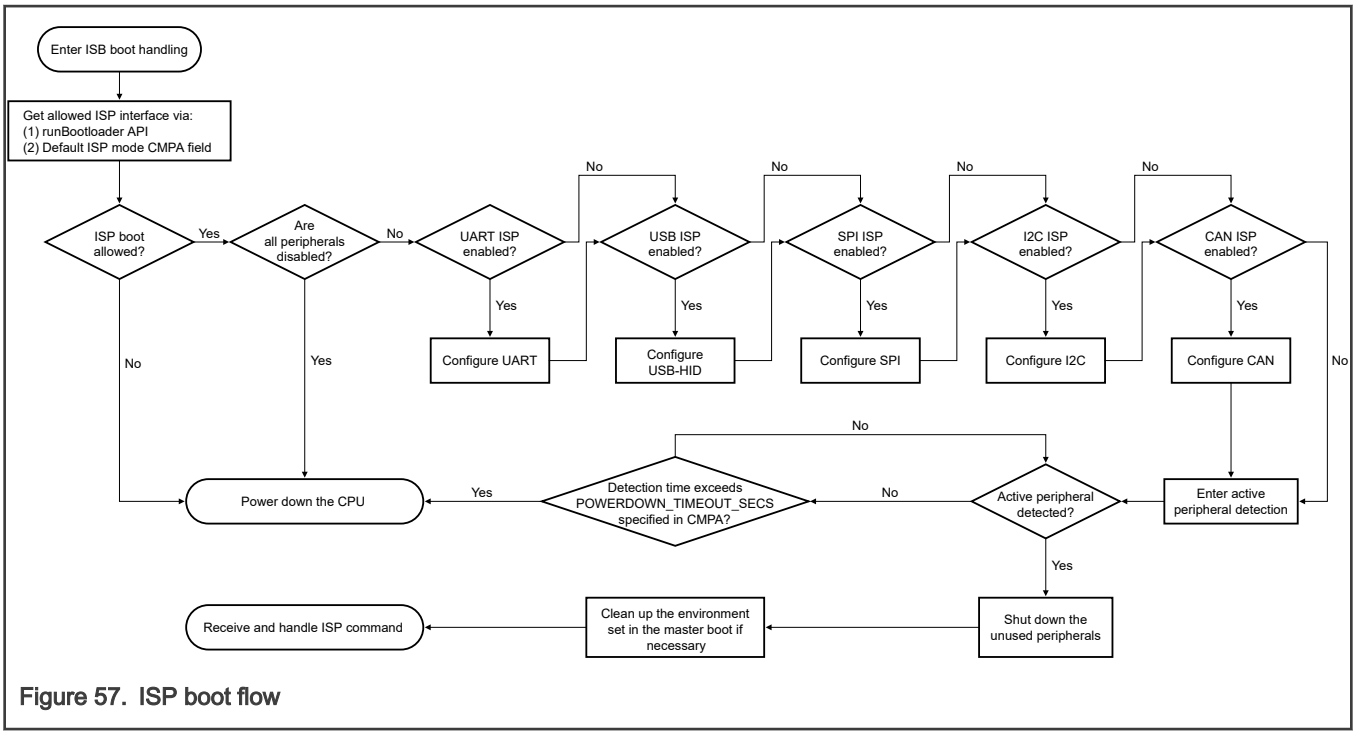


Figure 57. ISP boot flow

14.3 ISP protocol

This section explains the general protocol for packet transfers between the host and the bootloader. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with an incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

Commands can include an optional data phase:

- If the data phase is incoming (from the host to the bootloader), it is part of the original command.
- If the data phase is outgoing (from the bootloader to host), it is part of the response command.

14.3.1 Command with no data phase

The protocol for a command with no data phase contains:

1. Command packet (from the host).

2. GenericResponse command packet (to host).

The ACK sent in response to a command or a data packet can arrive at any time before, during, or after the command or data packet has been processed. See the diagrams in the Bootloader command set section.

14.3.2 Command with incoming data phase

The protocol for a command with incoming data phase contains:

1. Command packet (from host) (kCommandFlag_HasDataPhase set).
2. GenericResponse command packet (to host).
3. Incoming data packets (from the host).
4. GenericResponse command packet to host.

Note:

- The host may not send any further packets while it is waiting for the response to a command.
- The data phase is aborted if the GenericResponse packet, prior to the start of the data phase, does not have a status of kStatus_Success.
- Data phases may be aborted by the receiving side by sending the final GenericResponse early with a status of kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final GenericResponse packet sent after the data phase includes the status of the entire operation.

14.3.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:

1. Command packet (from the host).
2. ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set).
3. Outgoing data packets (to host).
4. GenericResponse command packet (to host).

- The data phase is considered part of the response command for the outgoing data phase sequence.
- The host may not send any further packets while the host is waiting for the response to a command.
- The data phase is aborted if the ReadMemory Response command packet, prior to the start of the data phase, does not contain the kCommandFlag_HasDataPhase flag.
- Data phases may be aborted by the device sending the final GenericResponse early with a status of kStatus_AbortDataPhase. The sending side may abort the data phase early by sending a zero-length data packet.
- The final GenericResponse packet sent after the data phase includes the status of the entire operation.

14.4 Bootloader packet types

14.4.1 Introduction

The bootloader device works in slave mode. All data communications are initiated by a host, which is either a PC or an embedded host. The bootloader device is the target, which receives a command or data packet. All data communications between host and target are packetized.

There are six types of packets used:

- Ping packet
- Ping response packet
- Framing packet

- Command packet
- Data packet
- Response packet

All fields in the packets are in little-endian byte order.

14.4.2 Ping packet

The ping packet is the first packet sent from a host to the target to establish a connection on the selected peripheral in order to run autobaud detection. The ping packet can be sent from host to target at any time that the target is expecting a command packet. If the selected peripheral is UART, a ping packet must be sent before any other communications. For other serial peripherals, it is optional.

In response to a ping packet, the target sends a ping response packet, discussed in the later sections.

Table 174. Ping packet format

Byte #	Value	Name
0	5Ah	Start byte
1	A6h	Ping

14.4.3 Ping response packet

The target sends a ping response packet back to the host after receiving a ping packet. If communication is over a UART peripheral, the target uses the incoming ping packet to determine the baud rate before replying with the ping response packet. After the ping response packet is received by the host, the connection is established, and the host starts sending commands to the target.

Table 175. Ping response packet format

Byte	Value	Parameter
0	5Ah	Start byte
1	A7h	Ping response code
2	00h	Protocol bugfix
3	03h	Protocol minor
4	01h	Protocol major
5	50h	Protocol name = 'P' (50h)
6	00h	Options low
7	00h	Options high
8	fbh	CRC16 low
9	40h	CRC16 high

To run autobaud for the UART peripheral, the ping response packet must be sent by the host when a connection is first established. For other serial peripherals, it is optional but recommended to determine the serial protocol version. The version number is in the same format as the bootloader version number returned by the GetProperty command.

14.4.4 Framing packet

The framing packet is used for flow control and error detection for the communications links that do not have such features built in. The framing packet structure sits between the link layer and the command layer. It wraps command and data packets as well.

Every framing packet containing data sent in one direction results in a synchronizing response framing packet in the opposite direction.

The framing packet described in this section is used for serial peripherals including the UART, CAN, I^2C, and SPI. The USB HID peripheral does not use framing packets. Instead, the packetization inherent in the USB protocol itself is used.

Table 176. Framing packet format

Byte	Value	Parameter	Description
0	5Ah	Start byte	—
1		PacketType	—
2		Length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes. —
3		Length_high	
4		Crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See CRC16 algorithm.
5		Crc16_high	
6....*n*		Command or Data packet payload	—

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

Table 177. Special framing packet format

Byte	Value	Parameter
0	5Ah	Start byte
1	A*n*h	packetType

The Packet Type field specifies the type of packet from one of the defined types (below):

Table 178. Packet type field

Packet type	Name	Description
A1h	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.

Table continues on the next page...

Table 178. Packet type field (continued)

Packet type	Name	Description
A2h	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
A3h	kFramingPacketType_AckAbort	Data phase is being aborted.
A4h	kFramingPacketType_Command	The framing packet contains a command packet payload.
A5h	kFramingPacketType_Data	The framing packet contains a data packet payload.
A6h	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.
A7h	kFramingPacketType_PingResponse	A response to ping. It contains the framing protocol version number and options.

14.4.5 CRC16 algorithm

The CRC is computed over each byte in the framing packet header, excluding the CRC16 field itself, and all of the payload bytes. The CRC algorithm is the XMODEM variant of CRC16.

The characteristics of the XMODEM variants are:

Table 179. CRC16 algorithm

Width	16
Polynomial	1021h
Init value	0000h
Reflect in	False
Reflect out	False
Xor out	0000h
Check result	31C3h

The check result is computed by running the ASCII character sequence "123456789" through the algorithm.

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
{
    uint32_t crc = 0;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
        {
            uint32_t i;
            uint32_t byte = src[j];
            crc ^= byte << 8;
            for (i = 0; i < 8; ++i)
                {
```

```
                    uint32_t temp = crc << 1;
                    if (crc & 8000h)
                        {
                            temp ^= 1021h;
                        }
                    crc = temp;
                }
        }
    return crc;
}
```

14.4.6 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 180. Command packet format

Command packet format (32 bytes)										
Command header (4 bytes)				28 bytes for Parameters (Max 7 parameters)						
Tag	Flags	Rsvd	Param Count	Param 1 (32-bit)	Param 2 (32-bit)	Param 3 (32-bit)	Param 4 (32-bit)	Param 5 (32-bit)	Param 6 (32-bit)	Param 7 (32-bit)

Table 181. Command header format

Byte #	Command header field	Description
0	Command or Response tag	The command header is 4 bytes long with these fields.
1	Flags	
2	Reserved. Must be 00h.	
3	ParameterCount	

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only seven parameters can fit into the command packet.

- **Flags:** Each command packet contains a flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets follow the command sequence. The number of bytes that are transferred in the data phase is determined by a command-specific parameter in the parameters array.
- **ParameterCount:** The number of parameters included in the command packet.
- **Parameters:** The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to seven parameters.

Command packets are also used by the target to send responses back to the host. As mentioned previously, command packets and data packets are embedded into framing packets for all transfers.

Table 182. Command tags

Command tag	Name	Description
01h	FlashEraseAll	The command tag specifies which command is supported by the bootloader. The valid command tags for the bootloader are listed here.[1]
02h	FlashEraseRegion	
03h	ReadMemory	
04h	WriteMemory	
05h	FillMemory	
06h	Reserved	
07h	GetProperty	
08h	ReceiveSbFile	
09h	Execute	
0Ah	Call	
0Bh	Reset	
0Ch	SetProperty	
0Dh	Reserved	
0Eh	eFuseProgram/FlashProgramOnce	
0Fh	eFuseRead/FlashReadOnce	
10h	Reserved	
11h	ConfigureMemory	
12h	Reserved	
13h	Reserved	
14h	Reserved	
15h	Reserved	
16h	TrustProvisioning	

1. The GetProperty, Reset, TrustProvisioning, SetProperty, and ReceiveSbFile are allowed in limited ISP mode.

Table 183. Response tags

Response tag	Name	Description
A0h	GenericResponse	The response tag specifies which of the responses the bootloader (target) returns to the host. The valid response tags are listed here.
A3h	ReadMemoryResponse	
A7h	GetPropertyResponse (used for sending responses to GetProperty command only)	
A3h	ReadMemoryResponse (used for sending responses to ReadMemory command only)	
AFh	FlashReadOnceResponse (used for sending responses to FlashReadOnce command only)	
B6h	TrustProvisioningResponse(used for sending response to TrustProvisioning command only)	

14.4.7 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse
- FlashReadOnceResponse
- TrustProvisioningResponse

GenericResponse: After the bootloader has processed a command, the bootloader sends a generic response with status and command tag information to the host. GenericResponse is the last packet in the command protocol sequence. The GenericResponse packet contains the framing packet data and the command packet data (with GenericResponse tag = A0h) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

Table 184. GenericResponse parameters

Byte #	Parameter	Description
0–3	Status code	Bootloader status error codes are errors encountered during the execution of a command by the target. If a command succeeds, then a kStatus_Success code is returned.
4–7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command or response tag set to a GetPropertyResponse tag value (A7h).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

Table 185. GetPropertyResponse parameters

Byte #	Value	Parameter
0–3		Status code
4–7		Property value
. . .		. . .
		There can be up to a maximum of 6 property values, limited to the size of the 32-bit command packet and property type.

ReadMemoryResponse: The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the command or response tag set to a ReadMemoryResponse tag value (A3h), and the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count is set to 2 for the status code and the data byte count parameters shown below.

Table 186. ReadMemoryResponse parameters

Byte #	Parameter	Description
0–3	Status code	The status of the associated Read Memory command.
4–7	Data byte count	The number of bytes sent in the data phase.

FlashReadOnceResponse: The FlashReadOnceResponse packet is sent by the target in response to the host sending a FlashReadOnce command. The FlashReadOnceResponse packet contains the framing packet data and the command packet data, with the commanresponse tag set to a FlashReadOnceResponse tag value (AFh), and the flags field set to 0. The parameter count is set to 2 plus the number of words requested to be read in the FlashReadOnceCommand.

Table 187. FlashReadOnceResponse parameters

Byte #	Value	Parameter
0–3		Status code
4–7		Byte count to read
. . .		. . .
		Can be up to 20 bytes of requested read data.

TrustProvisioningResponse: The TrustProvisioningResponse packet is sent by the target in response to the host sending a TrustProvisioning command. The TrustProvisioningResponse packet contains the framing packet data and command packet data, with the command or response tag set to a TrustProvisioningResponse tag value (B6h), and the flags field set to 0. The parameter count varies for different operations.

14.5 Bootloader command set

14.5.1 Introduction

All bootloader commands follow the command packet format wrapped by the framing packet as explained in previous sections.

For a list of status codes returned by bootloader see Bootloader status error codes.

14.5.2 GetProperty command

The GetProperty command is used to query the bootloader about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag occupies the first parameter of the command packet. The target returns a GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

The 32-bit property tag is the only parameter required for GetProperty command.

Table 188. Parameters for GetProperty Command

Byte #	Parameter
0 - 3	Property tag (which can be received from the GetProperty 0 command)
4 - 7	External Memory Identifier (only applies to GetProperty for external memory, or status identifier if the property tag is equal to 8).

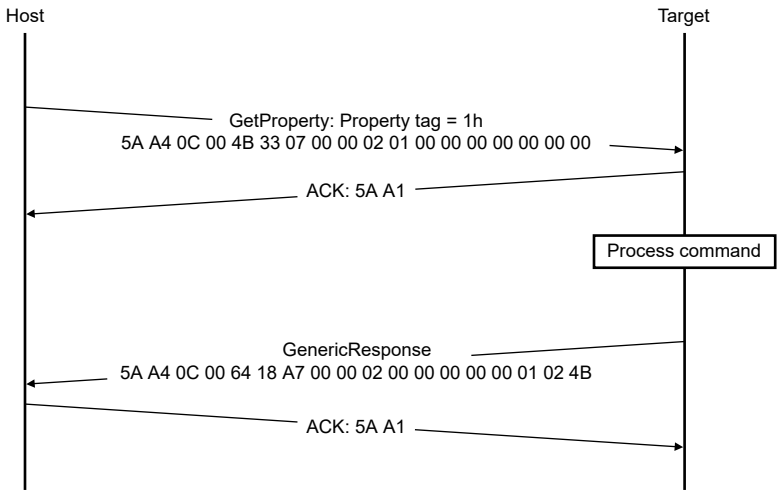


Figure 58. Protocol sequence for GetProperty command

Table 189. GetProperty command packet format (example)

GetProperty	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	0Ch 00h
	Crc16	4Bh 33h

Table 190. GetProperty command packet format (example)

GetProperty	Parameter	Value
Command packet	CommandTag	07h—GetProperty
	Flags	00h
	Reserved	00h
	ParameterCount	02h
	PropertyTag	0000_0001h—CurrentVersion
	Memory ID	0000_0000h—Internal Flash

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target sends a GetPropertyResponse packet with the response tag set to A7h. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). Table 191 shows an example of a GetProperty response packet.

Table 191. GetProperty response packet format (example)

GetPropertyResponse	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	0Ch 00h (12 bytes)
	Crc16	07h 7Ah
Command packet	ResponseTag	A7h
	Flags	00h
	Reserved	00h
	ParameterCount	02h
	Status	0000_0000h
	PropertyValue	0000_014Bh—CurrentVersion

14.5.3 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options of the bootloader. The command accepts the same property tags used with the GetProperty command. However, only some properties are writable. If an attempt to write a read-only property is made, an error is returned indicating the property is read-only and cannot be changed.

The property tag and the new value to set are the two parameters required for the SetProperty command.

Table 192. Parameters for SetProperty command

Byte #	Command
0—3	Property tag
4—7	Property value

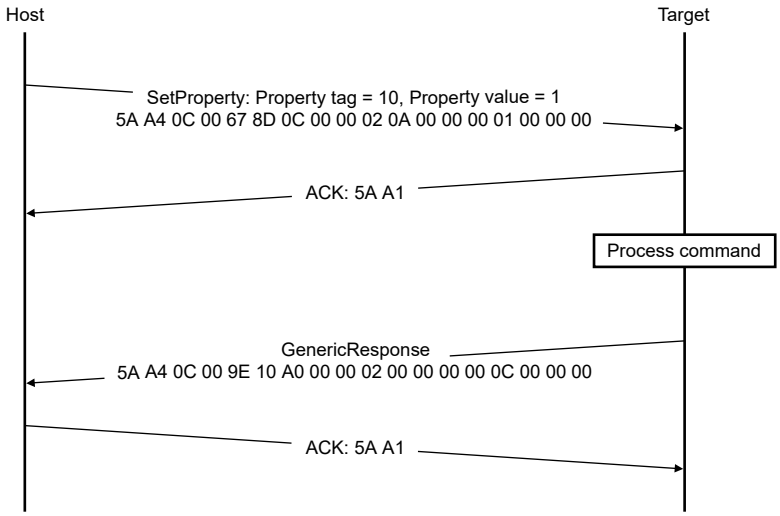


Figure 59. Protocol sequence for SetProperty Command

Table 193. SetProperty command packet format (example)

SetProperty	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	0Ch 00h
	Crc16	67h 8Dh
Command packet	CommandTag	0Ch—SetProperty with property tag 10
	Flags	00h
	Reserved	00h
	ParameterCount	02h
	PropertyTag	0000_000Ah—VerifyWrites
	PropertyValue	0000_0001h

The SetProperty command has no data phase.

Response: The target returns a GenericResponse packet with one of the following status codes:

Table 194. SetProperty response status codes

Status code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

14.5.4 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory (excluding IFR region). If any flash regions are protected, then the FlashEraseAll command fails and returns an error status code. The command tag for FlashEraseAll command is 01h set in the commandTag field of the command packet.

The FlashEraseAll command requires memory ID. If memory ID is not specified, the internal flash (memory ID = 0) is selected as default.

Table 195. Parameter for FlashEraseAll command

Byte #	Parameter	Value	Description
0-3	Memory ID	000h	Internal Flash
		09h	FlexSPI NOR
		110h	Serial NOR / EEPROM through SPI

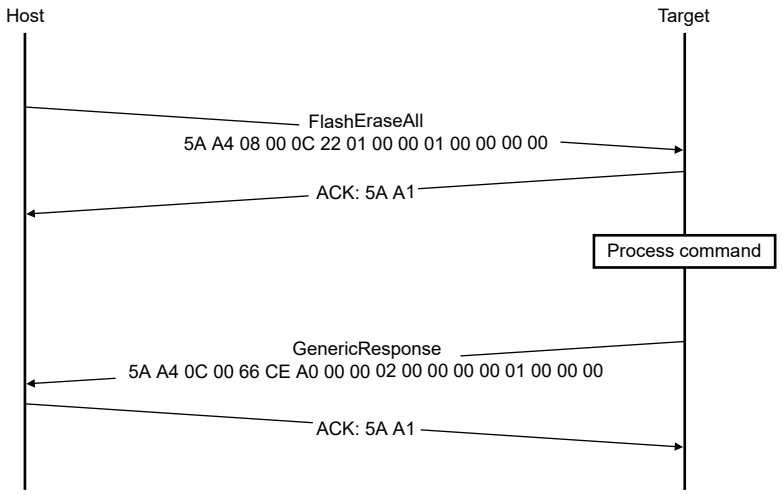


Figure 60. FlashEraseAll command

Table 196. FlashEraseAll command packet format (example)

FlashEraseAll	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	08h 00h
	Crc16	0Ch 22h
Command packet	CommandTag	01h—FlashEraseAll
	Flags	00h
	Reserved	00h
	ParameterCount	01h
	Memory ID	See the above table.

The FlashEraseAll command has no data phase.

Response: The target returns a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command or set to an appropriate error status code.

14.5.5 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory.

The start address and number of bytes are the two parameters required for the FlashEraseRegion command. The start address and byte count parameters must be 4-byte aligned ([1:0] = 00), or the FlashEraseRegion command fails and returns kStatus_FlashAlignmentError (101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command fails and returns kStatus_FlashAddressError (102). If any part of the region specified is protected, the FlashEraseRegion command fails and returns kStatus_MemoryRangeInvalid (10200).

Table 197. Parameter for FlashEraseRegion command

Byte #	Parameter
0–3	Start address
4–7	Byte count
8–11	Memory ID

The FlashEraseRegion command has no data phase.

Response: The target returns a GenericResponse packet with one of the following error status codes.

Table 198. FlashEraseRegion response status codes

Status code
kStatus_Success (0).

Table continues on the next page...

Table 198. FlashEraseRegion response status codes (continued)

Status code
kStatus_MemoryRangeInvalid (10200).
kStatus_FlashAlignmentError (101).
kStatus_IFlashAddressError (102).
kStatus_FlashAccessError (103).
kStatus_FlashProtectionViolation (104).
kStatus_FlashCommandFailure (105).

14.5.6 ReadMemory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory that is accessible by the CPU and not protected by security.

The start address and the number of bytes are the two parameters required for the ReadMemory command. The memory ID is optional. Internal memory is selected as the default if memory ID is not specified.

Table 199. Parameter for read memory command

Byte #	Parameter	Description
0–3	Start address	Start address of memory to read from
4–7	Byte count	Number of bytes to read and return to the caller
8–11	Memory ID	Internal or external memory identifier

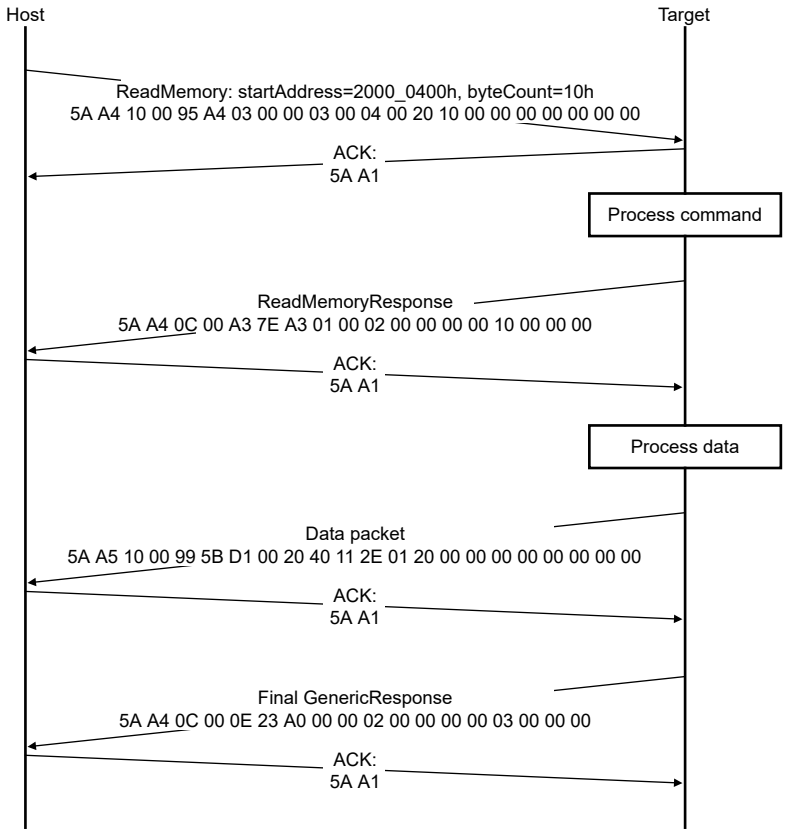


Figure 61. Command sequence for ReadMemory

Table 200. ReadMemory command packet format (example)

ReadMemory	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	10h 00h
	Crc16	F4h 1Bh
Command packet	CommandTag	03h—ReadMemory
	Flags	00h
	Reserved	00h
	ParameterCount	03h
	StartAddress	2000_0400h
	ByteCount	0000_0064h
	Memory ID	0h

Data phase: The ReadMemory command has a data phase. Because the target works in slave mode, the host needs to pull data packets until it has received the number of bytes of data specified in the byteCount parameter of the ReadMemory command.

Response: The target returns a GenericResponse packet with a status code either set to kStatus_Success (upon successful execution of the command) or set to an appropriate error status code.

14.5.7 WriteMemory command

The WriteMemory command writes data that is provided in the data phase to a specified range of bytes in memory (flash memory or RAM). However, if flash memory protection is enabled, then writes to protected sectors fail.

Consider the following information when you write to flash memory:

- First, any flash sector you write to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash memory requires the start address to be page-aligned.
- The byte count is rounded up to a page size, and trailing bytes are filled with the flash erase pattern (FFh).
- If the VerifyWrites property is set to true, then writes to flash memory also perform a flash memory verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

The two parameters required for the WriteMemory command are the start address and the number of bytes. The memory ID is optional. Internal memory is selected as the default if memory ID is not specified.

Table 201. Parameters for WriteMemory command

Byte #	Command
0–3	Start address
4–7	Byte count
8–11	Memory ID

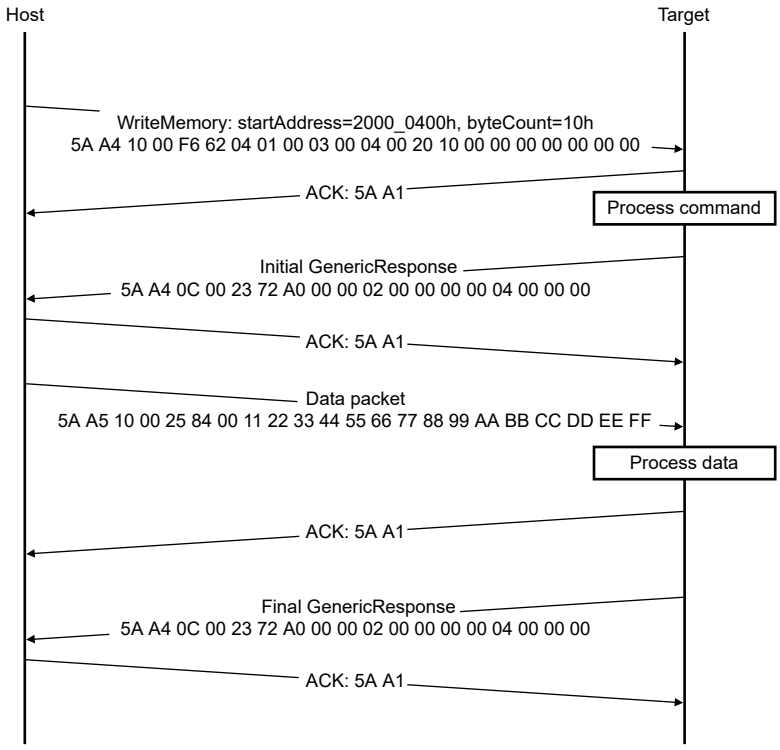


Figure 62. Protocol sequence for WriteMemory command

Table 202. WriteMemory command packet format (example)

WriteMemory	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	10h 00h
	Crc16	97h DDh
Command packet	CommandTag	04—WriteMemory
	Flags	01h
	Reserved	00h
	ParameterCount	03h
	StartAddress	2000_0400h
	ByteCount	0000_0064h
	Memory ID	0h

Data phase: The WriteMemory command has a data phase: the host sends data packets until the target has received the number of bytes of data specified in the byteCount parameter of the WriteMemory command.

Response: The target returns a GenericResponse packet with a status code set to kStatus_Success (upon successful execution of the command), or to an appropriate error status code.

14.5.8 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that the FillMemory command parameter includes a data pattern. Also, there is no data phase for the FillMemory command, but WriteMemory does have a data phase.

Table 203. Parameters for FillMemory command

Byte #	Command
0–3	Start address of memory to fill
4–7	Number of bytes to write with the pattern • The start address must be 32-bit aligned. • The number of bytes must be evenly divisible by 4.
8–11	32-bit pattern

- To fill with a byte pattern (8-bit), the byte must be replicated four times in the 32-bit pattern.
- To fill with a short pattern (16-bit), the short value must be replicated two times in the 32-bit pattern.

For example, to fill a byte value with FEh, the word pattern is FEFE_FEFEh; to fill a short value 5AFEh, the word pattern is 5AFE_5AFEh.

Consider the following information when you write to flash memory:

- First, any flash memory sector you write to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash memory requires the start address to be 4-byte aligned ([1:0] = 00).
- If the VerifyWrites property is set to true, then writes to flash memory also perform a flash memory verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

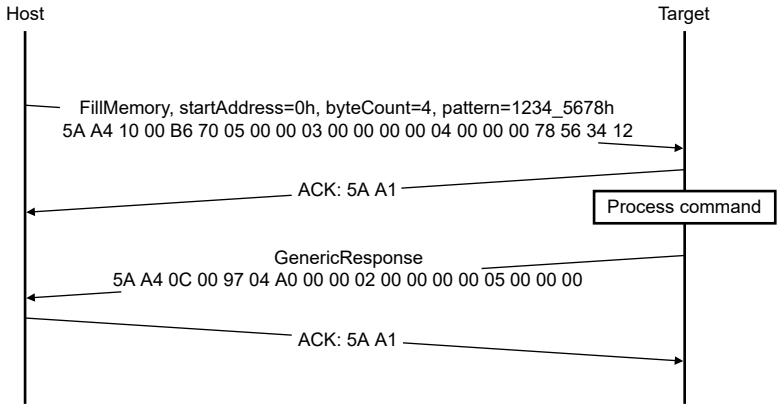


Figure 63. Protocol sequence for FillMemory command

Table 204. FillMemory command packet format (example)

FillMemory	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	10h 00h
	Crc16	E4h 57h
Command packet	CommandTag	05h—FillMemory
	Flags	01h
	Reserved	00h
	ParameterCount	03h
	StartAddress	0000_7000h
	ByteCount	0000_0800h
	PatternWord	1234_5678h

The FillMemory command has no data phase.

Response: The target (bootloader) returns a GenericResponse packet with a status code set to kStatus_Success (upon successful execution of the command), or to an appropriate error status code.

14.5.9 Execute command

The Execute command results in the bootloader setting of each of these items:

- The program counter to the code at the provided jump address
- R0 to the provided argument
- A stack pointer to the provided stack pointer address

Prior to the jump, the system returns to the Reset state.

The Execute command requires these parameters:

- Jump address
- Function argument pointer
- Stack pointer

If the stack pointer is set to zero, the called code is responsible for setting the processor stack pointer before using the stack.

Table 205. Parameters for Execute command

Byte #	Command
0–3	Jump address

Table continues on the next page...

Table 205. Parameters for Execute command (continued)

Byte #	Command
4–7	Argument word
8–11	Stack pointer address

The Execute command has no data phase.

Response: Before executing the Execute command, the target validates the parameters and returns a GenericResponse packet with a status code, set to either kStatus_Success or to an appropriate error status code.

14.5.10 Reset command

The Reset command results in the bootloader resetting the chip.

The Reset command requires no parameters.

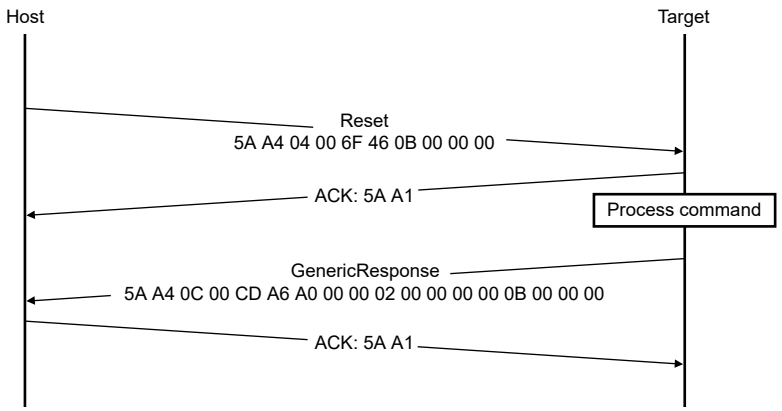


Figure 64. Protocol sequence for Reset command

Table 206. Reset command packet format (example)

Reset	Parameter	Value
Framing packet	Start byte	5Ah
	PacketType	A4h, kFramingPacketType_Command
	Length	04h 00h
	Crc16	6Fh 46h
Command packet	CommandTag	0Bh—reset
	Flags	00h
	Reserved	00h
	ParameterCount	03h

The Reset command has no data phase.

Response: Before resetting the chip, the target returns a GenericResponse packet with status code set to kStatus_Success.

You can also use the Reset command to switch boot from flash memory after successful flash image provisioning via the ROM bootloader. After issuing the Reset command, allow five seconds for the user application to start running from flash memory.

14.5.11 eFuseProgramOnce/FlashProgramOnce command

The FlashProgramOnce command writes data (provided in a command packet) to a specified range of bytes in the program-once field.

Consider the following information when you write to the program-once field:

- The program-once field only supports programming once, so any attempt to reprogram a program-once field generates an error response.
- Writing to the program-once field requires the byte count to be 4-byte aligned or 8-byte aligned.

The FlashProgramOnce command uses three parameters:

- Index 2
- byteCount
- Data

Table 207. Parameters for FlashProgramOnce command

Byte #	Command
0–3	Index of program-once field
4–7	Byte count (must be evenly divisible by 4)
8–11	Data

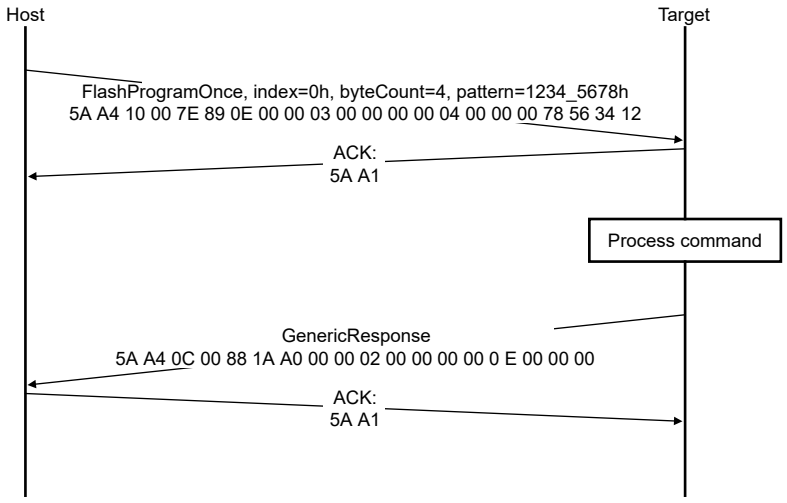


Figure 65. Protocol sequence for FlashProgramOnce command

Table 208. FlashProgramOnce command packet format

FlashProgramOnce	Parameter	Value
Framing packet	start byte	5Ah
	packetType	A4h, kFramingPacketType_Command
	length	10h 00h
	crc16	7E4h 89h
Command packet	commandTag	0Eh—FlashProgramOnce
	flags	0
	reserved	0
	parameterCount	3
	index	0000_0051h (the customer last efuse index)
	byteCount	0000_0004h
	Data	1234_5678h

Response: The target (MCX bootloader) returns a GenericResponse packet with a status code set to kStatus_Success (upon successful execution of the command), or to an appropriate error status code.

14.5.12 eFuseReadOnce / FlashReadOnce command

The FlashReadOnce command returns the contents of the program-once field by providing the given index and byte count.

The FlashReadOnce command uses two parameters: Index and byteCount. The eFuseReadOnce command uses only one parameter: Index.

Table 209. Parameters for FlashReadOnce command

Byte #	Parameter	Description
0–3	Index	Index of the program-once field (to read from)
4–7	byteCount	Number of bytes to read and return to the caller

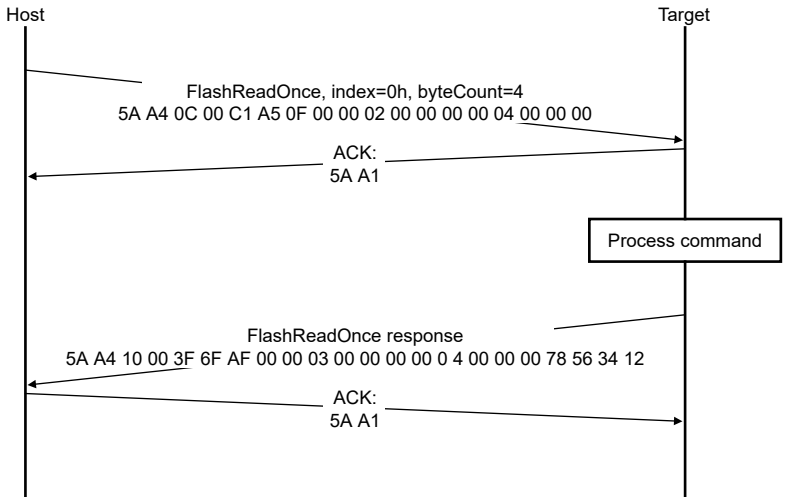


Figure 66. Protocol sequence for FlashReadOnce command

Table 210. FlashReadOnce command packet format

FlashReadOnce	Parameter	Value
Framing packet	start byte	5Ah
	packetType	A4h
	Length	0Ch 00h
	Crc	C1h A5h
Command packet	commandTag	0Fh—FlashReadOnce
	Flags	00h
	Reserved	00h
	parameterCount	02h
	Index	0000_0000h
	byteCount	0000_0004h

Table 211. FlashReadOnce response format

FlashReadOnce response	Parameter	Value
Framing packet	start byte	5Ah
	packetType	A4h
	Length	10h 00h

Table continues on the next page...

Table 211. FlashReadOnce response format (continued)

FlashReadOnce response	Parameter	Value
	CRC	3Fh 6Fh
Command packet	commandTag	AFh
	Flags	00h
	Reserved	00h
	parameterCount	03h
	Status	0000_0000h
	byteCount	0000_0004h
	Data	1234_5678h

Response: Upon successful execution of the command, the target returns:

- A FlashReadOnceResponse packet with a status code set to kStatus_Success
- A byte count
- A corresponding data read from the program-once field

If the execution was not successful, then the target returns a status code set to an appropriate error status and a byte count set to 0.

14.5.13 ConfigureMemory command

The ConfigureMemory command configures an internal or external memory device using a preprogrammed configuration block. The parameters passed in the command are:

- The memory ID
- The memory address from which the configuration data can be loaded

One case for loading the data could be a scenario where the configuration data is written to a RAM or flash memory location. This command would then direct the bootloader to use the data at that location for configuration of the external memory device.

Table 212. Parameters for ConfigureMemory command

Byte #	Command
0–3	Memory ID
4–7	Configuration block address

Response: The target (bootloader) returns a GenericResponse packet with a status code either set to kStatus_Success (upon successful execution of the command) or set to an appropriate error code.

The following table shows the supported memory IDs.

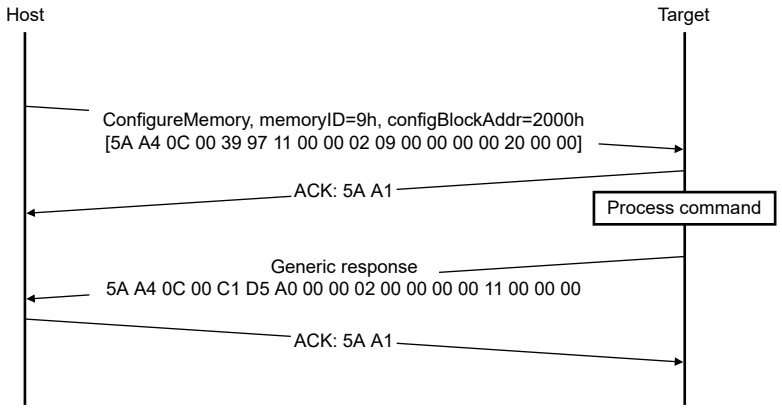


Figure 67. Protocol sequence for ConfigureMemory command

Table 213. Supported memory IDs

Memory ID	Description
0	Internal RAM/FLASH (used for the PRINCE configuration).
9h	External FlexSPI device. See Serial NOR Flash through FlexSPI.
110h	External 1-bit SPI NOR FLASH device. See 1-bit Serial NOR flash through SPI.

14.5.14 ReceiveSBFile command

The Receive SB File command (ReceiveSbFile) starts the transfer of an SB file to the target. This command only specifies the size in bytes of the SB file that is sent in the data phase. The SB file is processed as it is received by the bootloader.

Table 214. Parameters for Receive SB File command

Byte #	Command
0–3	Byte count

Data phase: The Receive SB file command has a data phase. The host sends data packets until the target has received the number of bytes specified in the byteCount parameter of the Receive SB File command.

Response: The target returns a GenericResponse packet with a status code set to the kStatus_Success (upon successful execution of the command) or set to an appropriate error code.

14.5.15 GetProperty/SetProperty command properties

This section lists the properties of the GetProperty and SetProperty commands.

Table 215. Properties used by GetProperty and SetProperty commands, sorted by value

Property	Writable	Tag value	Size	Description
CurrentVersion	No	01h	4	Current bootloader version
AvailablePeripherals	No	02h	4	The set of peripherals supported on this chip

Table continues on the next page...

Table 215. Properties used by GetProperty and SetProperty commands, sorted by value (continued)

Property	Writable	Tag value	Size	Description
FlashStartAddress	No	03h	4	Start address of program flash memory
FlashSizeInBytes	No	04h	4	Size in bytes of program flash memory
AvailableCommands	No	07h	4	The set of commands supported by the bootloader
Check Status	No	08h	4	Return the status based on the specified status identifier 0—CRC status 32-bit return value for CRC check 10401—Application CRC check failed 10402—Application CRC check is inactive 10403—Application CRC check invalid 1—Last Error See details of the last error in GetLastError property.
MaxPacketSize	No	0Bh	4	Maximum supported packet size for currently active peripheral interface.
ReservedRegions	No	0Ch	8 x *n*	List of memory regions reserved by bootloader. Returned as value pairs (<start-address-of-region>,<end-address-of-region>). If HasDataPhase flag is not set, then the response packet parameter count indicates the number of pairs. If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase. *n* indicates the number of memory region pairs.
SystemDeviceId	No	10h	4	Value from SYSCON's Device ID (DEVICE_ID0) and Chip Revision ID and Number (DIEID) registers.
LifeCycleState	No	11h	4	The life cycle of the device. 5AA5_5AA5h—Device is in development life cycle C33C_C33Ch—Device is in deployment life cycle
UniqueDevice/UUID	No	12h	16	Unique device identification.
Target ROM Version	No	18h	4	Target build version number.
ExternalMemoryAttributes	No	19h	24	List of attributes supported by the specified memory ID (110h = SPINOR FLASH). See description of the return value in ExternalMemoryAttributes property.
IrqNotifierPin	Yes	1ch	4	IRQ Notifier Pin setting: • bit[7:0]: GPIO pin • bit[15:8]: GPIO port • bit [30:16]: Reserved

Table continues on the next page...

Table 215. Properties used by GetProperty and SetProperty commands, sorted by value (continued)

Property	Writable	Tag value	Size	Description
				• bit[31]: Enable flag 0: disable, 1: enable

14.5.15.1 Property definitions

14.5.15.1.1 CurrentVersion property

The value of this property is a 4-byte structure containing the current version of the bootloader.

Table 216. CurrentVersion property fields

Bit	[31:24]	[23:16]	[15:8]	[7:0]
Field	Name = 'K' (4Bh)	Major version	Major version	Bugfix version

14.5.15.1.2 AvailablePeripherals property

The value of this property is the peripherals supported by the bootloader, and the hardware on which it runs.

Table 217. Peripheral bits

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Field	Reserved	Reserved	Reserved	USB HID	CAN	SPI slave	I2C slave	LPUART

If the peripheral is available, then the corresponding bit is set in the property value. All reserved bits must = 0.

14.5.15.1.3 AvailableCommands property

This property value is a bit field that indicates the commands enabled in the bootloader. Only commands that can be sent from the host to the target are listed in the bit field. Response commands such as GenericResponse are excluded.

The bit number that identifies whether a command is present is the command's tag value minus 1. The value 1 is subtracted from the command tag because the lowest command tag value is 01h. To determine the bit mask for a given command, use this expression: mask = 1 << (tag – 1).

Table 218. Command bits

Bit	[Others]	[20]	[16]	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Command																			

14.5.15.1.4 ExternalMemoryAttributes property

The value that this property returns is a 24-byte data structure containing available external memory attributes:

- Start address
- Total size (in KB)
- Page size
- Sector size
- Block size

Below is the breakdown of the 24-byte structure.

Table 219. Fields of ExternalMemoryAttributes property

Field offset	Field Description
0–3	The value returned is a bitmap showing the supported attributes for the external memory, with the corresponding bit field set. 0000_0001h—start address 0000_0002h—total size 0000_0004h—page size 0000_0008h—sector size 0000_0010h—block size
4–7	Start address of external memory
8–11	Total size of external memory in KB
12–15	Page size of external memory in bytes
16–19	Sector size of external memory in bytes
20–23	Block size of external memory in bytes

14.5.15.1.5 GetLastError property

The following table lists the response words and corresponding error conditions.

Table 220. Response word and error description

Response word	Error	Description
0B37_F300h	kLog_Auth_ImageTypeCheck_Fail	When checking the image type pass, the image type did not meet the authentication condition.
0B32_F300h	kLog_Auth_ReadKeyStore_Fail	The read of the keystore part from the RAM image or another keystore region failed.
0B34_F300h	kLog_Auth_VerifyHMAC_Fail	Failed to verify the mac key.
0B35_F300h	kLog_Auth_ImageEntryCheck_Fail	Application entry point is invalid or stack address is invalid.
0B36_F300h	kLog_Auth_Auth_Fail	The authentication of the image failed.
0B36_F301h	kLog_Auth_Auth_Fail_Time	The authentication failed, and the cost time has been loaded into the log data.
0B37_F300h	kLog_Auth_CrcCheck_Fail	CRC checksum is wrong.
0B37_F301h	kLog_Auth_CrcCheck_Fail_Time	The image CRC check failed and the cost time has been loaded into the log data.
0B37_FC00h	kLog_Auth_CrcCheck_Invalid	Either the image type or the image length = 0 did not meet the requirement of the CRC image.
0B38_F300h	kLog_Auth_Dice_Fail	Dice calculation failed.
0C00_F300h	kLog_Jump_Fail	The image jump failed.

Table 221. Response word and error description

Response word	Error	Description
0C00_F500h	kLog_Jump_Fail_Fatal	Failed to jump to application.
0702_F301h	kLog_Recoveryboot_Fail_Reason	Recovery boot failed.
0501_F300h	kLog_Masterboot_Init_Fail	The master boot initialization failed.
0801_F300h	kLog_Ispboot_Init_Fail	The ISP boot initialization failed, meaning there are no available ISP peripherals.
0901_F300h	kLog_BootDevice_Init_Fail	The boot device initialization failed.
0902_F300h	kLog_BootDevice_Read_Fail	The master boot failed to read the initial image.
0A20_F300h	kLog_ImgLoad_InitLoad_Fail	The master boot failed to load the image header part.
0A21_F300h	kLog_ImgLoad_RemainingLoad_Fail	Image header and configuration data loaded successfully; the remaining portion of the image failed to load.
0B30_DC00h	kLog_Auth_SecureBootDisabled	The secure boot has been disabled—no need to do authentication.

14.6 Bootloader status error codes

Table 222 describes the status error codes that the bootloader returns to the host.

Table 222. Bootloader status error codes, sorted by value

Error code	Value	Description
kStatus_Success	0	Operation succeeded without error
kStatus_Fail	1	Operation failed with generic error
kStatus_ReadOnly	2	Requested value is read-only and cannot be changed
kStatus_OutOfRange	3	Requested value is out of range
kStatus_InvalidArgument	4	Requested command's argument is undefined
kStatus_Timeout	5	Timeout occurred
kStatus_NoTransferInProgress	6	No send in progress
kStatus_FLASH_Success	0	API executed successfully
kStatus_FLASH_InvalidArgument	4	Invalid argument was provided
kStatus_FlashSizeError	100	Not used
kStatus_FlashAlignmentError	101	Address or length does not meet required alignment

Table continues on the next page...

Table 222. Bootloader status error codes, sorted by value (continued)

Error code	Value	Description
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FLASH_CommandFailure	105	INT_STATUS[FAIL] = 1
kStatus_FlashUnknownProperty	106	Unknown flash memory property
kStatus_FlashEraseKeyError	107	Provided key does not match programmed flash memory key
kStatus_FlashRegionExecuteOnly	108	Area of flash memory is protected as execute-only
kStatus_FLASH_ExecuteInRamFunctionNotReady	109	Execute-in-RAM function is not available
kStatus_FLASH_CommandNotSupported	111	Flash memory API is not supported
kStatus_FLASH_ReadOnlyProperty	112	Flash memory property is read-only
kStatus_FLASH_InvalidPropertyValue	113	Flash memory property value out of range
kStatus_FLASH_InvalidSpeculationOption	114	Flash memory prefetch speculation option is invalid
kStatus_FLASH_EccError	116	A correctable or uncorrectable error occurred during command execution
kStatus_FLASH_CompareError	117	Destination and source memory contents do not match
kStatus_FLASH_RegulationLoss	118	Loss of regulation during read
kStatus_FLASH_InvalidWaitStateCycles	119	Wait state cycle set to read/write mode is invalid
kStatus_FLASH_OutOfDateCfpaPage	132	CFPA page version is out of date
kStatus_FLASH_BlankIfrPageData	133	Blank page cannot be read
kStatus_FLASH_EncryptedRegionsEraseNotDoneAtOnce	134	Encrypted flash memory subregions not erased at once
kStatus_FLASH_ProgramVerificationNotAllowed	135	Program verification not allowed when encryption enabled
kStatus_FLASH_HashCheckError	136	Hash check of page data failed
kStatus_FLASH_SealedFfrRegion	137	FFR region sealed

Table continues on the next page...

Table 222. Bootloader status error codes, sorted by value (continued)

Error code	Value	Description
kStatus_FLASH_FfrRegionWriteBroken	138	FFR spec region not allowed to be written discontinuously
kStatus_FLASH_NmpaAccessNotAllowed	139	NMPA region not allowed to be read, written, or erased
kStatus_FLASH_CmpaCfgDirectEraseNotAllowed	140	CMPA configuration region cannot be erased directly
kStatus_FLASH_FfrBankIsLocked	141	FFR bank region locked
kStatus_FLASH_CfpaScratchPageInvalid	148	CFPA scratch page invalid
kStatus_FLASH_CfpaVersionRollbackDisallowed	149	CFPA version rollback not allowed
kStatus_FLASH_ReadHidingAreaDisallowed	150	Flash memory hiding read not allowed
kStatus_FLASH_ModifyProtectedAreaDisallowed	151	Flash firewall page locked Erase and program are not allowed
kStatus_FLASH_CommandOperationInProgress	152	Flash memory state busy Flash memory command is in progress
kStatus_FLASH_IFR1_AccessNotAllowed	153	Access out of the IFR flash region
kStatus_FLASH_EraseVerifyFailed	154	The flash verify erase fail, indicate that flash is not erased
kStatus_UnknownCommand	10000	Command not recognized
kStatus_SecurityViolation	10001	Security violation happened when receiving disallowed commands
kStatus_AbortDataPhase	10002	Sender requested data phase termination
kStatus_Ping	10003	Ping command received from host
kStatus_NoResponse	10004	No response from host
kStatus_NoResponseExpected	10005	Expected no response from host
kStatus_CommandUnsupported	10006	Unsupported command received
kStatusRomLdrSectionOverrun	10100	Reached the end of SB file processing
kStatusRomLdrSignature	10101	Signature or version is incorrect
kStatusRomLdrSectionLength	10102	The bootOffset or new section count is out of range

Table continues on the next page...

Table 222. Bootloader status error codes, sorted by value (continued)

Error code	Value	Description
kStatusRomLdrUnencryptedOnly	10103	Unencrypted image disabled
kStatusRomLdrEOFReached	10104	End of image file has been reached
kStatusRomLdrChecksum	10105	Checksum of command tag block is invalid
kStatusRomLdrCrc32Error	10106	CRC-32 of the data for a load command is incorrect
kStatusRomLdrUnknownCommand	10107	Unknown command was found in SB file
kStatusRomLdrIdNotFound	10108	No bootable section found in SB file
kStatusRomLdrDataUnderrun	10109	SB state machine waiting for more data
kStatusRomLdrJumpReturned	10110	The function jumped to by the SB file has returned
kStatusRomLdrCallFailed	10111	Call command in SB file failed
kStatusRomLdrKeyNotFound	10112	Matching key to unencrypt the section not found in SB file's key dictionary
kStatusRomLdrSecureOnly	10113	SB file sent is unencrypted Security on the target is enabled
kStatusRomLdrResetReturned	10114	SB file reset operation has unexpectedly returned
kStatusRomLdrRollbackBlocked	10115	Image version rollback event detected
kStatusRomLdrInvalidSectionMacCount	10116	Invalid section MAC count detected in SB file
kStatusRomLdrUnexpectedCommand	10117	Command tag in SB file is unexpected
kStatusRomLdrBadSBKEK	10118	Bad SBKEK detected
kStatusRomLdrPendingJumpCommand	10119	Jump command pending Actual jump implemented in sbloader_finalize()
kStatusMemoryRangeInvalid	10200	Requested address range does not match an entry, or the length extends past the matching entry's end address
kStatusMemoryReadFailed	10201	Memory read failed
kStatusMemoryWriteFailed	10202	Memory write failed

Table continues on the next page...

Table 222. Bootloader status error codes, sorted by value (continued)

Error code	Value	Description
kStatusMemoryCumulativeWrite	10203	Cumulative write occurred due to a write to an unerased flash memory region
kStatusMemoryNotConfigured	10205	Memory was not configured before access
kStatusMemoryAlignmentError	10206	Alignment error during memory access
kStatusMemoryVerifyFailed	10207	Verifying operation failed after erasing or programming flash memory
kStatusMemoryWriteProtected	10208	Memory to be written is protected
kStatusMemoryAddressError	10209	Memory address is invalid or wrong
kStatusMemoryBlankCheckFailed	10210	Check of blank memory failed
kStatusMemoryBlankPageReadDisallowed	10211	Memory is blank Read command is disallowed
kStatusMemoryProtectedPageReadDisallowed	10212	Memory is protected Read command is disallowed
kStatusMemoryFfrSpecRegionWriteBroken	10213	Write operation to FFR region broken
kStatusMemoryUnsupportedCommand	10214	Memory command not supported
kStatus_UnknownProperty	10300	Requested property value undefined
kStatus_ReadOnlyProperty	10301	Requested property value cannot be written
kStatus_InvalidPropertyValue	10302	Specified property value invalid
kStatus_AppCrcCheckPassed	10400	CRC check valid and passed
kStatus_AppCrcCheckFailed	10401	CRC check valid but failed
kStatus_AppCrcCheckInactive	10402	CRC check inactive
kStatus_AppCrcCheckInvalid	10403	CRC check invalid because BCA invalid, or CRC parameters unset (all FFh bytes)
kStatus_AppCrcCheckOutOfRange	10404	CRC check valid, but addresses out of range
kStatus_RomApiExecuteCompleted	0	ROM successfully processed complete SB file or boot image
kStatus_RomApiNeedMoreData	10801	ROM needs more data to continue processing boot image

Table continues on the next page...

Table 222. Bootloader status error codes, sorted by value (continued)

Error code	Value	Description
kStatus_RomApiBufferSizeNotEnough	10802	User buffer not large enough for use by Kboot during execution
kStatus_RomApiInvalidBuffer	10803	User buffer not acceptable for sbloader or authentication
kStatus_FLEXSPI_SequenceExecutionTimeout	6000	FLEXSPI sequence execution timed out
kStatus_FLEXSPI_InvalidSequence	6001	FLEXSPI LUT sequence invalid
kStatus_FLEXSPI_DeviceTimeout	6002	FLEXSPI device timed out
kStatus_FLEXSPINOR_ProgramFail	20100	Page programming failure
kStatus_FLEXSPINOR_EraseSectorFail	20101	Sector erase failure
kStatus_FLEXSPINOR_EraseAllFail	20102	Chip erase failure
kStatus_FLEXSPINOR_WaitTimeout	20103	Execution timeout
kStatus_FlexSPINOR_NotSupported	20104	Page size overflow
kStatus_FlexSPINOR_WriteAlignmentError	20105	Address alignment error
kStatus_FlexSPINOR_CommandFailure	20106	Erase or Program verify error
kStatus_FlexSPINOR_SFDP_NotFound	20107	Timeout occurred during the API call
kStatus_FLEXSPINOR_Unsupported_SFDP_Version	20108	Unrecognized SFDP version
kStatus_FLEXSPINOR_FLASH_NotFound	20109	Flash memory detection failure
kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	20110	DDR read dummy probe failure
kStatus_IAP_Success	0	IAP API execution succeeded
kStatus_IAP_Fail	1	IAP API execution failed
kStatus_IAP_InvalidArgument	100001	Invalid argument detected during API execution
kStatus_IAP_OutOfMemory	100002	Heap size not large enough during API execution
kStatus_IAP_ReadDisallowed	100003	Read memory operation disallowed during API execution
kStatus_IAP_CumulativeWrite	100004	Flash memory region to be programmed is not empty

Table continues on the next page...

Table 222. Bootloader status error codes, sorted by value (continued)

Error code	Value	Description
kStatus_IAP_EraseFailuire	100005	Erase operation failed
kStatus_IAP_CommandNotSupported	100006	Specific command not supported
kStatus_IAP_MemoryAccessDisabled	100007	Memory access disabled **NOTE** Typically occurs on FLEXSPI NOR if it is not configured properly
kStatus_NBOOT_Success	5A5A_5A5Ahu	Operation successful in NBOOT functions
kStatus_NBOOT_Fail	5A5A_A5A5hu	Operation failed in NBOOT functions
kStatus_NBOOT_InvalidArgument	5A5A_A5F0hu	Invalid argument passed to function in NBOOT functions
kStatus_NBOOT_RequestTimeout	5A5A_A5E1hu	Operation timed out in NBOOT functions.
kStatus_NBOOT_KeyNotLoaded	5A5A_A5E2hu	Requested key not loaded in NBOOT functions
kStatus_NBOOT_AuthFail	5A5A_A5E4hu	Authentication failed in NBOOT functions
kStatus_NBOOT_OperationNotAvaialable	5A5A_A5E5hu	Operation not available on this hardware in NBOOT functions
kStatus_NBOOT_KeyNotAvailable	5A5A_A5E6hu	Key not available in NBOOT functions
kStatus_NBOOT_IvCounterOverflow	5A5A_A5E7hu	Overflow of IV counter (PRINCE/IPED) in NBOOT functions
kStatus_NBOOT_SelftestFail	5A5A_A5E8hu	FIPS self-test failure in NBOOT functions
kStatus_NBOOT_InvalidDataFormat	5A5A_A5E9hu	Invalid data format for example antipole in NBOOT functions
kStatus_NBOOT_IskCertUserDataTooBig	5A5A_A5EAhu	Size of user data in ISK certificate greater than 96 bytes in NBOOT functions
kStatus_NBOOT_IskCertSignatureOffsetTooSmall	5A5A_A5EBhu	Signature offset in ISK certificate smaller than expected in NBOOT functions
kStatus_NBOOT_MemcpyFail	5A5A_845Ahu	Unexpected error detected during nboot_memcpy() in NBOOT functions
kStatus_SKBOOT_Success	5AC3_C35Ahu	Operation successful in SKBOOT functions
kStatus_SKBOOT_Fail	C35A_C35Ahu	Operation failed in SKBOOT functions
kStatus_SKBOOT_InvalidArgument	C35A_5AC3hu	Return invalid argument status in SKBOOT functions
kStatus_MKBOOT_Success	0000_C35Ahu	Operation successful in master KBOOT
kStatus_MKBOOT_Fail	0000_C3C3hu	Operation failed in master KBOOT.

Table continues on the next page...

Table 222. Bootloader status error codes, sorted by value (continued)

Error code	Value	Description
kStatus_MKBOOT_InvalidArgument	0000_5AC3hu	Return Invalid argument status in Master KBOOT

NOTE

In UART, I2C, CAN, and SPI ISP modes, the ROM expects posted responses to be read by the host within (20 ms x number of bytes)—otherwise the ROM reports the terminate command. Because an ACK from the ROM is only two bytes, the host must read those bytes within 40 ms of the ROM posting.

14.7 UART ISP

14.7.1 Introduction

The bootloader integrates an autobaud detection algorithm for the UART peripheral, thereby providing flexible baud rate choices.

Autobaud feature: If you use UART*n* to connect to the bootloader, then you must keep the UART*n*_RX pin high and not left floating during the detection phase. You perform this action to comply with the autobaud detection algorithm. After the bootloader detects the ping packet (5Ah A6h) on UART*n*_RX, the bootloader firmware executes the autobaud sequence.

If the bootloader successfully detects the baud rate, then it sends a ping packet response [(5Ah A7h), protocol version (4 bytes), protocol version options (2 bytes), and crc16 (2 bytes)] at the detected baud rate. The bootloader then enters a loop, waiting for bootloader commands via the UART peripheral.

NOTE

The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed UART transmission mode (8-bit data, no parity bit, and 1 stop bit). If the bytes of the ping packet are sent one by one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this instance, the autobaud detection state machine should be reset.

The baud rate is closely related to the chip core and system clock frequencies. The typical supported baud rates are:

- 9600
- 19200
- 38400
- 57600
- 115200
- 230400
- 460800
- 1000000

Packet transfer: After autobaud detection succeeds, bootloader communications can take place over the UART peripheral:

- Figure 68 shows how the host detects an ACK from the target.
- Figure 69 shows how the host detects a ping response from the target
- Figure 70 shows how the host detects a command response from the target

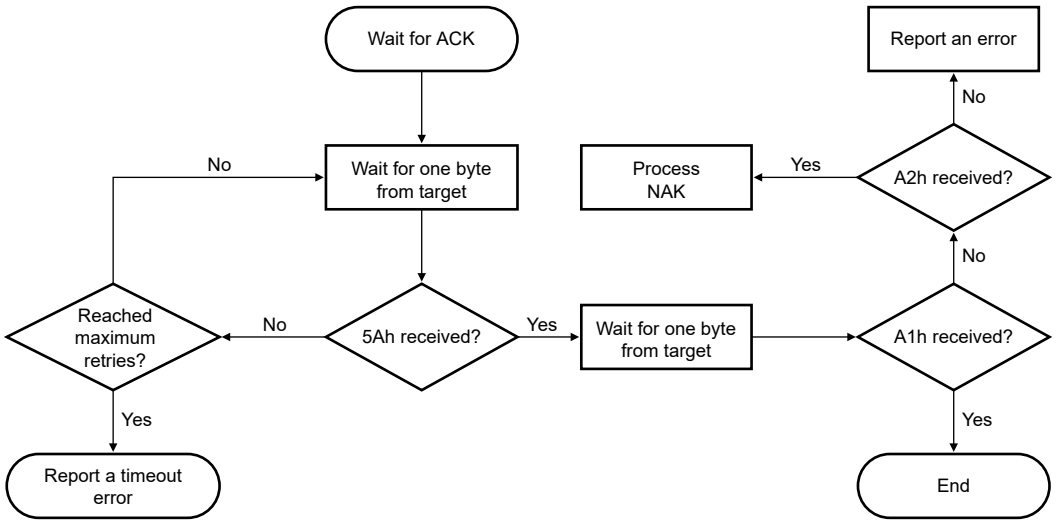


Figure 68. Host reads an ACK from target via UART

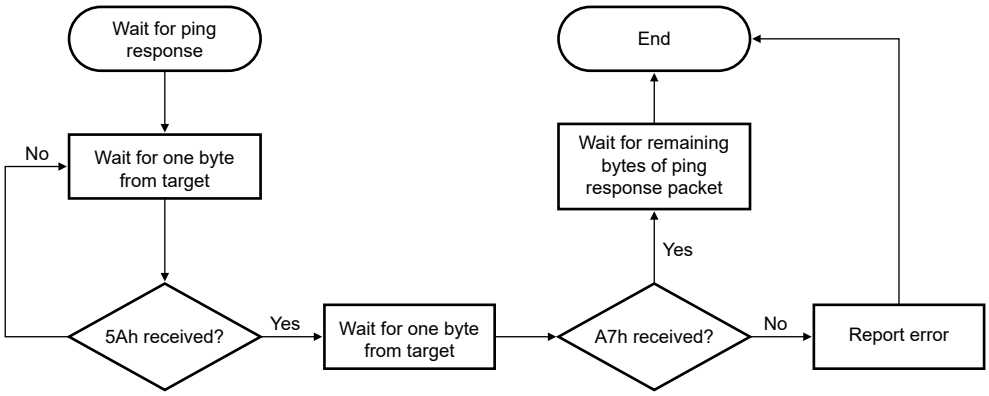


Figure 69. Host reads a ping response from target via UART

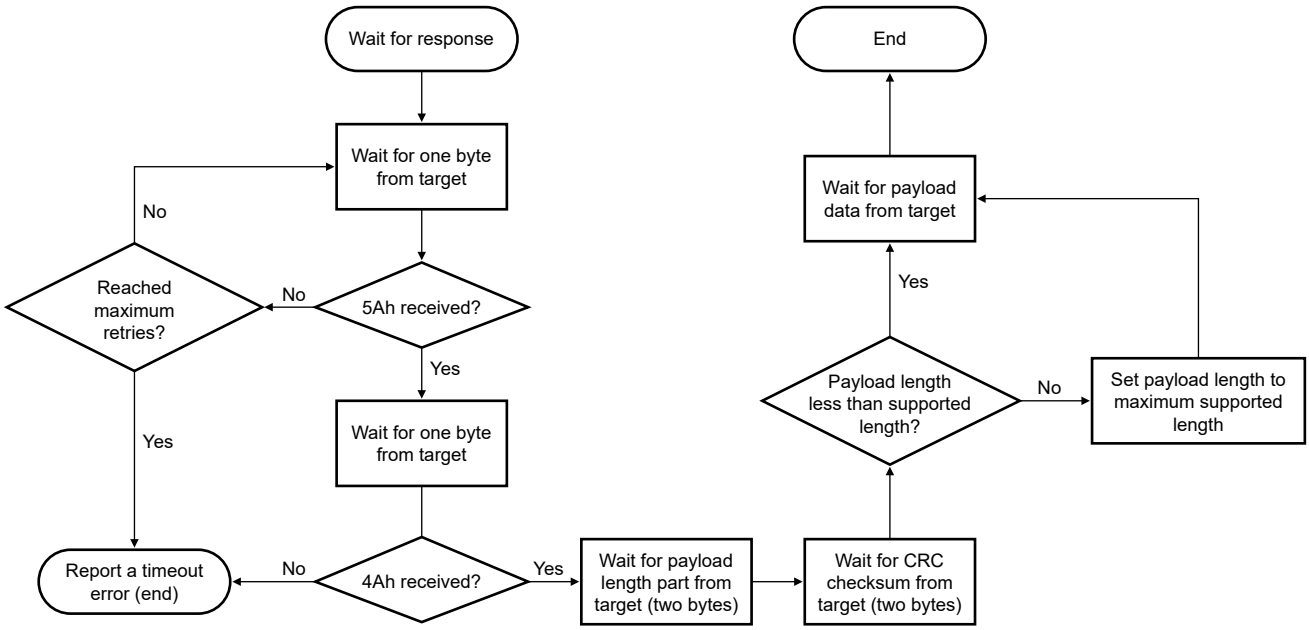


Figure 70. Host reads a command response from target via UART

For more information on UART ISP command, response, and data formats, see Bootloader packet types. For information on UART ISP commands, see Command packet.

14.8 I²C ISP

14.8.1 Introduction

The bootloader supports loading data into flash memory via the I²C peripheral, where the I²C peripheral serves as the I²C target. The transfer uses a 7-bit target address. The bootloader uses 10h as the I²C target address and supports up to 400 kbit/s as the I²C baud rate. You can also reconfigure the I2C slave address with the value from CMPA[I2C_SLAVE_ADDR](0x0100403C[7:0]).

The maximum supported I²C baud rate depends on the core clock frequency when the bootloader is running. The typical baud rate is 400 kbit/s with factory settings.

Because the I²C peripheral serves as an I²C target device, the host must start each transfer and fetch each outgoing packet:

- The host sends an incoming packet with a selected I²C target address, and the direction bit is set as write.
- The host reads an outgoing packet with a selected I²C target address, and the direction bit is set as read.
- If the target is busy with processing or preparing data, then the target sends 00h as the response to the host.

The following flowcharts show the communication flow of the host reading the ping and ACK packets, and the corresponding responses from the target.

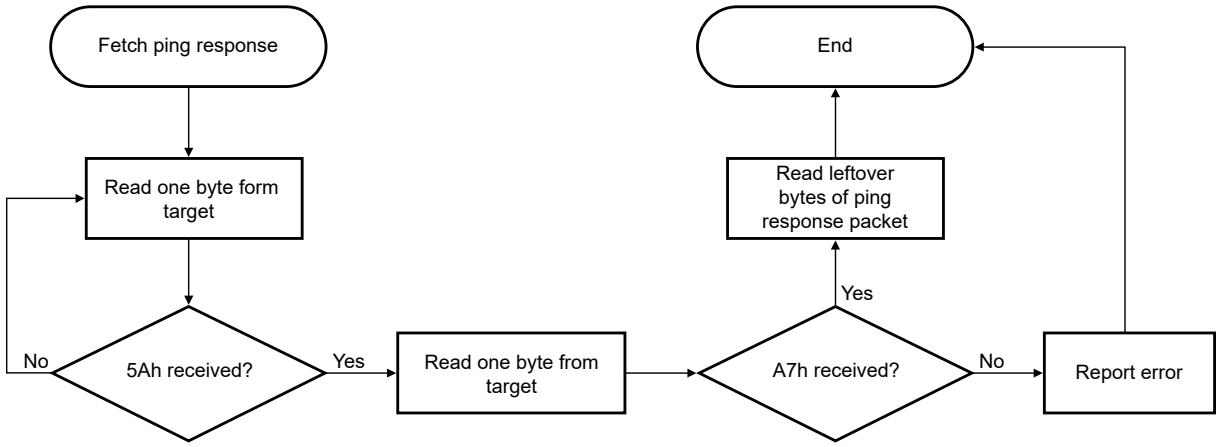


Figure 71. Host reads ping response from target via I²C

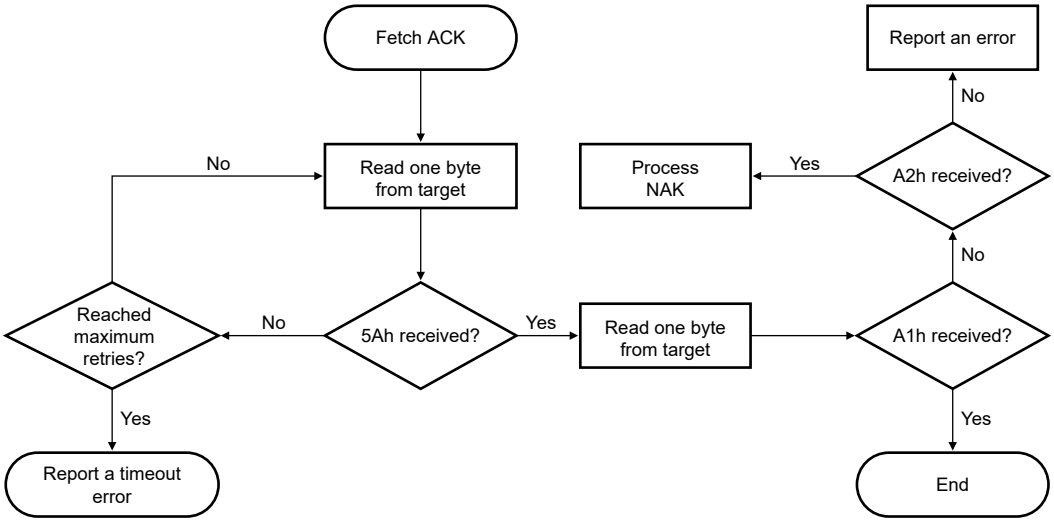


Figure 72. Host reads ACK packet from target via I²C

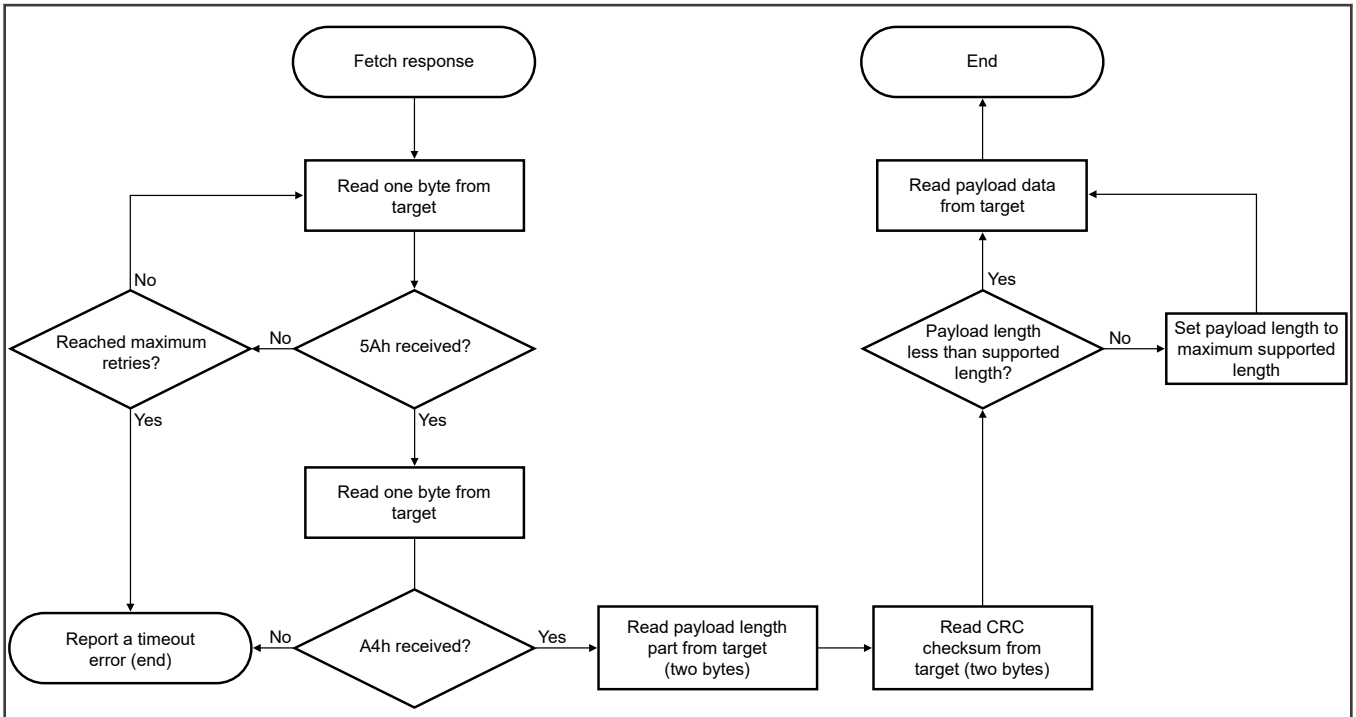


Figure 73. Host reads response from target via I²C

For more information on I²C ISP command, response, and data formats, see Bootloader packet types.

For more information on I²C ISP commands, see Bootloader command set.

14.9 SPI ISP

14.9.1 Introduction

The bootloader supports loading data into flash memory via the SPI peripheral, where the SPI peripheral serves as an SPI target. The SPI transfer should be in SPI mode 3 with 8 data bits.

The maximum supported baud rate of the SPI depends on the core clock frequency when the bootloader is running. The typical baud rate is 2000 kbit/s with the factory settings. The actual baud rate is lower or higher than 2000 kbit/s, depending on the actual value of the core clock.

Because the SPI peripheral serves as an SPI target device, the host must start each transfer and must fetch each outgoing packet.

The transfer on SPI is slightly different from I²C:

- The host receives 1 byte after it sends out any byte.
- Received bytes must be ignored when the host is sending out bytes to the target.
- The host starts reading bytes by sending 00h to the target.

The target sends the byte 00h as a response to the host if the target is under the following conditions:

- Processing incoming packet
- Preparing outgoing data
- Received invalid data

The bootloader also supports the active notification pin (nIRQ pin) to notify the host processor that it is busy or ready for new commands or data.

To accelerate the SPI transfer between the host and the bootloader, the bootloader provides the nIRQ pin, which you enable with the SetProperty command. After you enable it, the host must wait until it sees a negative edge on the nIRQ pin before reading any data from the bootloader. It must also wait until the nIRQ pin is high before sending any data to the bootloader.

See the figure below for the typical physical connection between the host and the bootloader device.

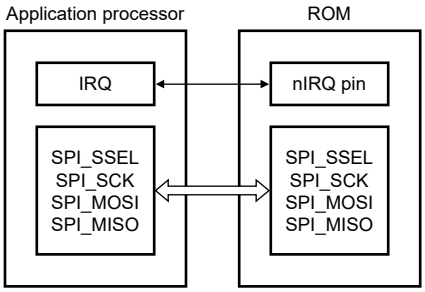


Figure 74. Physical interface for SPI ISP

The following flowcharts show how the host reads a ping response, an ACK, and a command response from the target via SPI, without the nIRQ pin enabled.

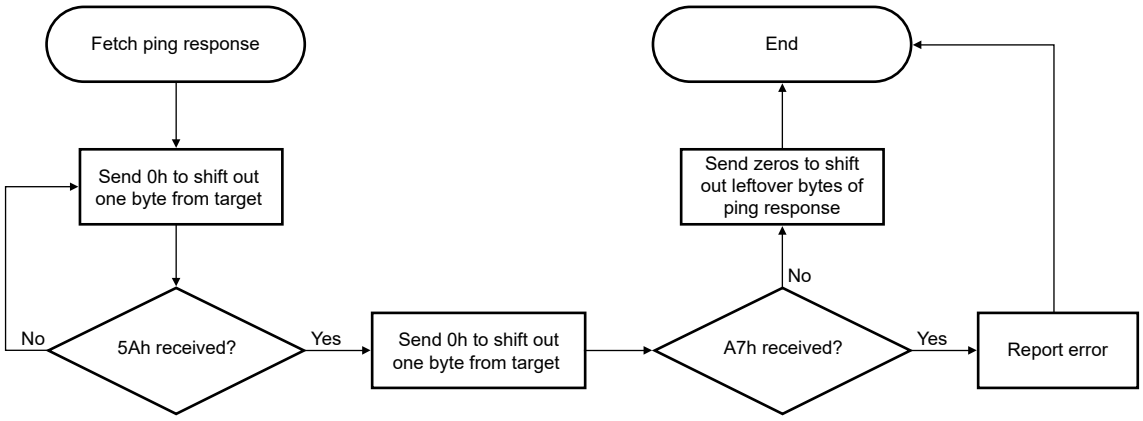


Figure 75. Host reads ping packet from target via SPI

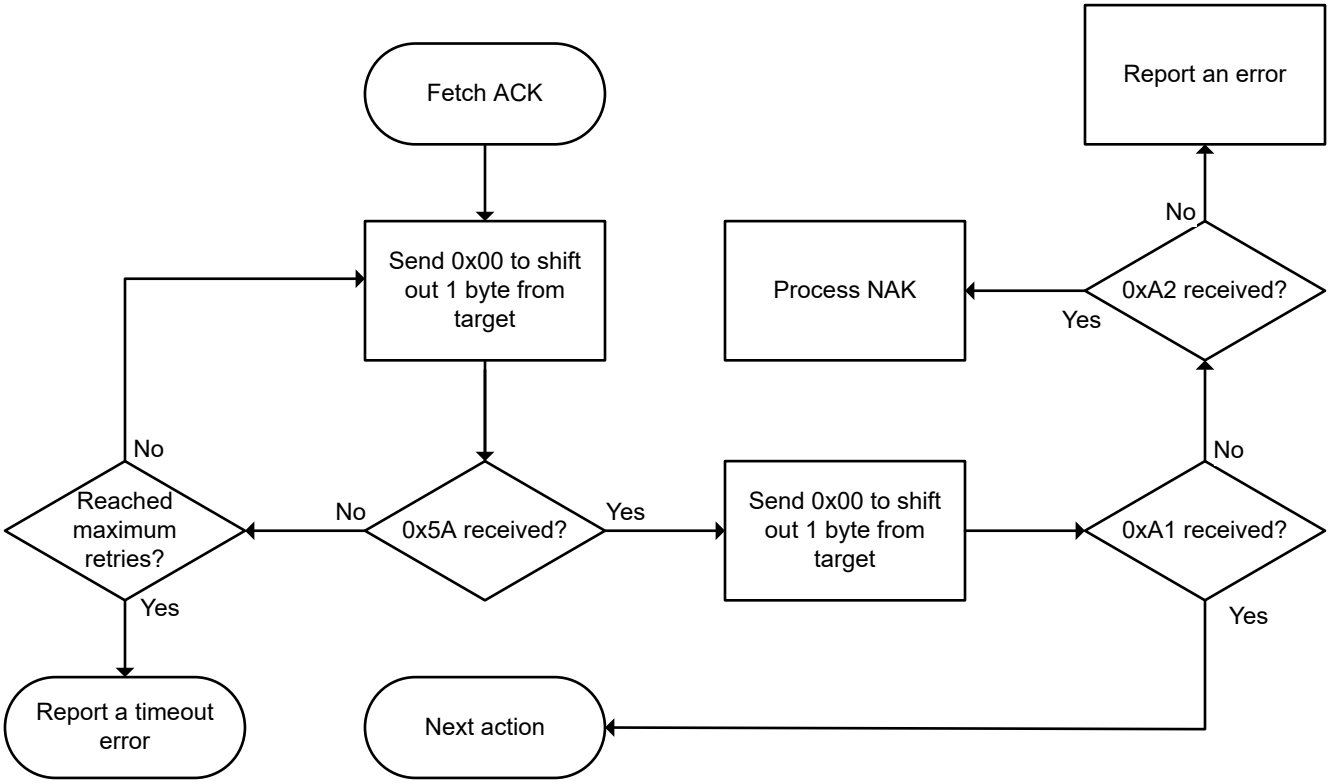


Figure 76. Host reads ACK from target via SPI

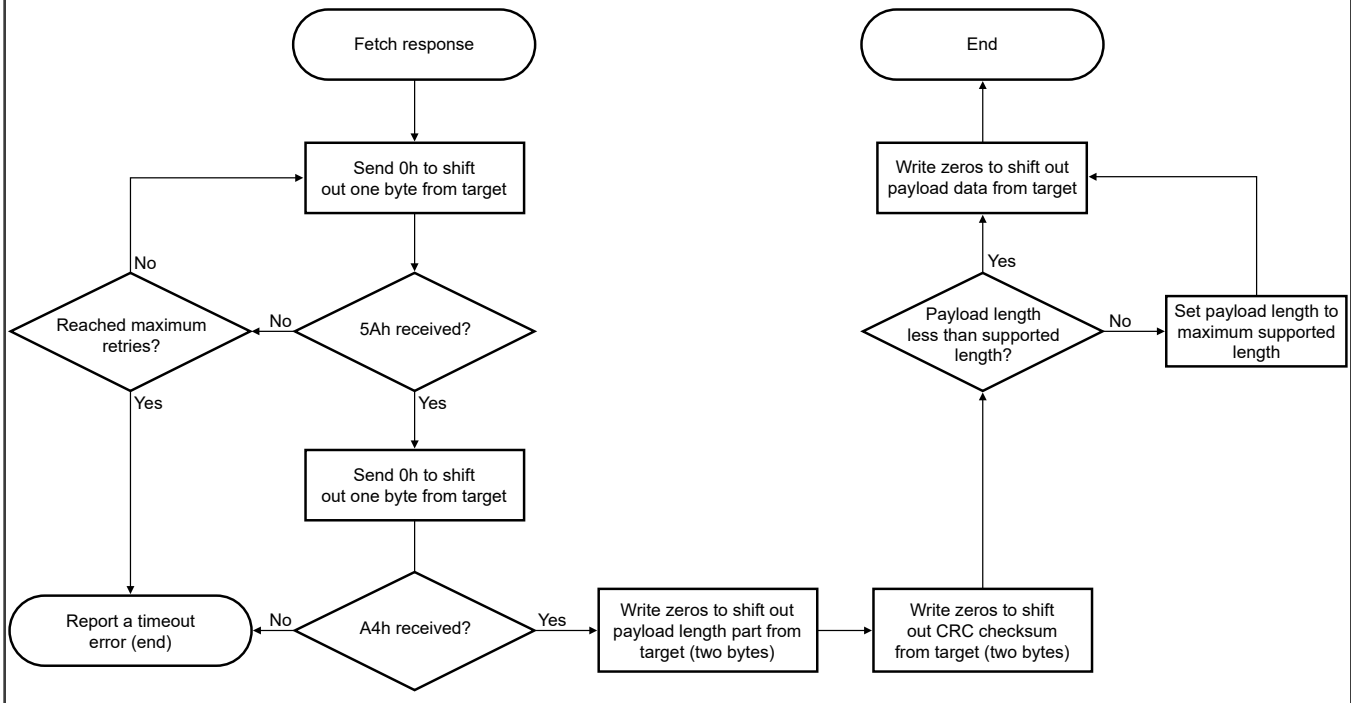


Figure 77. Host reads response from target via SPI

For more information on the command, response, and data formats of SPI ISP, see Bootloader packet types.

For more information on SPI ISP commands, see Bootloader command set.

14.9.2 Host read ACK(5Ah, A1h) packet timing restriction

If the ISP protocol does not use the IRQ pin, then the host must use the data-fetching timing for ACK from the chip:

- After sending out the command, the host must wait for at least 100 µs before polling the start byte 5Ah of the ACK packet sent by the chip. If the fetched data is not 5Ah, delay 100 µs and poll to reread 5Ah.
- For the ACK packet (5Ah, A1h): After the host receives 5Ah, you must add a 50 µs delay before fetching the A1h from the chip.

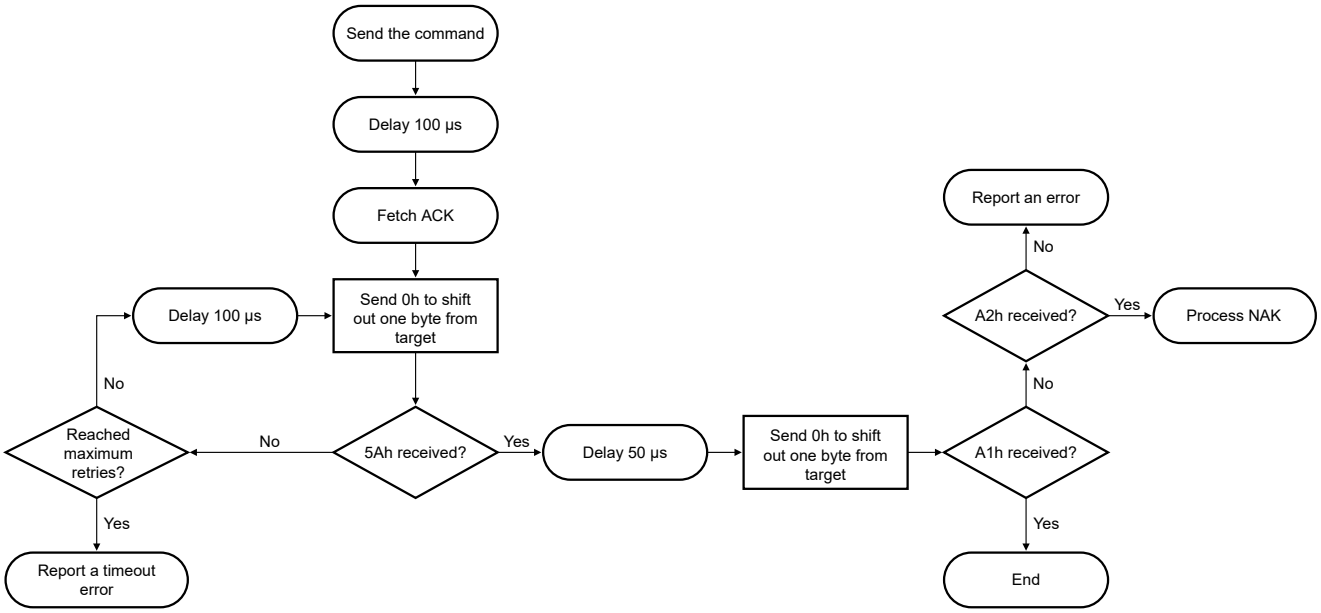


Figure 78. Host read ACK(5Ah, A1h) packet timing restriction flow

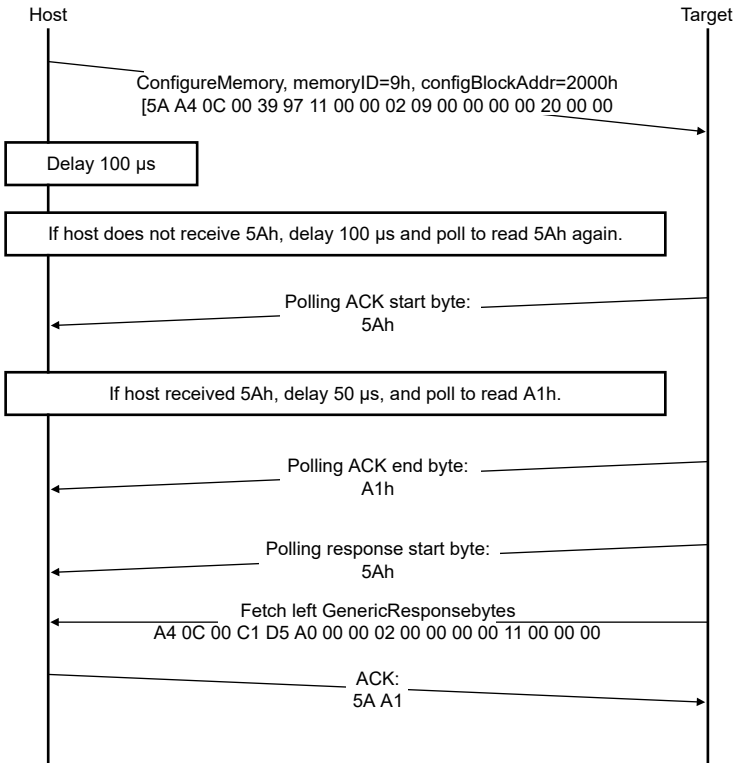


Figure 79. Host read ACK(5Ah, A1h) packet timing restriction

14.10 USB ISP

14.10.1 Introduction

The bootloader supports ISP using the USB peripheral. The target is implemented as a USB human interface device (HID) class.

USB HID does not use framing packets when transferring data through the USB HID class. Instead, USB HID uses the packetization inherent in the USB protocol itself. The chip's ability to NAK out transfers (until they can be received) provides the required flow control. The built-in CRC of each USB packet provides the required error detection.

14.10.1.1 Device descriptor

The bootloader configures the default USB VID, PID, and strings as follows:

Default VID and PID:

- VID = 1FC9h
- PID = 014fh

Default strings:

- Manufacturer [1] = "NXP SEMICONDUCTOR INC"
- Product [2] = "USB COMPOSITE DEVICE"

You can customize the USB VID, PID, and strings using the CMPA of the flash memory. For example, you can customize the USB VID and PID by writing:

- The new VID to the CMPA location 0100_4034h in flash memory
- The new PID to the usbPid field of the CMPA in flash memory

14.10.1.2 Endpoints

The HID peripheral uses three endpoints:

- Control (0)
- Interrupt IN (1)
- Interrupt OUT (2)

The interrupt OUT endpoint is optional for HID class devices. The chip bootloader uses the interrupt OUT endpoint as a pipe, where the firmware can NAK send requests from the USB host.

14.10.1.3 HID Reports

The bootloader USB HID peripheral defines and uses four HID reports. The report ID determines the direction and type of packet sent in the report—otherwise, the contents of all reports are the same.

Table 223. HID reports assigned for the bootloader

Report ID	Packet type	Direction
1	Command	OUT
2	Data	OUT
3	Command	IN
4	Data	IN

Each report has a maximum size of 60 bytes. The maximum payload size is 56 bytes. In addition, there is a 4-byte report header that indicates the length (in bytes) of the payload and a report ID sent to the packet.

The actual data sent in all reports uses the following format:

Table 224. Data format sent in USB HID packet

Report ID	Data
0	Report ID
1	Padding
2	Packet length LSB
3	Packet length MSB
4	Packet[0]
5	Packet[1]
6	Packet[2]
...	...
N+4-1	Packet[N-1]

If the HID report descriptor defines more than one report, the report ID must be included in the data. The actual data sent and received has a maximum length of 35 bytes. The packet length header is in little-endian format and it is set to the size (in bytes)

of the packet sent in the report. This size does not include the report ID or the packet length header itself. During a data phase, a packet size of 0 indicates a data phase abort request from the receiver.

For more information on USB ISP command and response formats, see Command packet.

For more information on USB ISP data format, see Response packet.

For more information on USB ISP commands, see Bootloader command set.

14.11 CAN ISP

The bootloader supports loading data into flash through the CAN peripheral. It supports four predefined auto-detect speeds on CAN transferring:

- 125 KHz
- 250 KHz
- 500 KHz
- 750 KHz
- 1 MHz

The current CAN IP can support up to 1 MHz using the ROM ISP feature, so the default baud rate is set to 1 MHz.

In host applications, you can specify the baud rate for CAN by writing one of the values in the following list to (0100_4028[31:28]) in CMPA:

- 0000: Auto baud detection (125 KHz, 250 KHz, 500 KHz, 750 KHz, 1 MHz)
- 0001: 125 kbit/s
- 0010: 250 kbit/s
- 0011: 500 kbit/s
- 0100: 750 kbit/s
- 0101 and above: 1000 kbit/s and auto baud detection

Initially, the bootloader enters the listen mode with the default speed of 1 MHz. After the host sends a ping to a specific node, it generates traffic on the CAN bus. Because the bootloader is in a listen mode, it can check whether the local node speed is correct by detecting errors. If there is an error, some traffic is visible, but it may not be at the right speed to read the correct data. If this happens, the speed setting changes and checks for errors again. No errors indicates that the speed is correct.

The settings change back to the normal receiving mode to check if there is a package for this node. It then stays in this speed until another host using another speed tries to communicate with any node. It repeats the process to detect a correct speed before sending host time-out and aborting the request.

The host side should have a reasonable time tolerance during the auto-speed detection period. If it sends a time-out signal, that indicates either there is no response from the specific node, or there is a real error and it must report the error to the application.

This flowchart demonstrates the communication flow for how the host reads the ping packet, ACK, and response from the target.

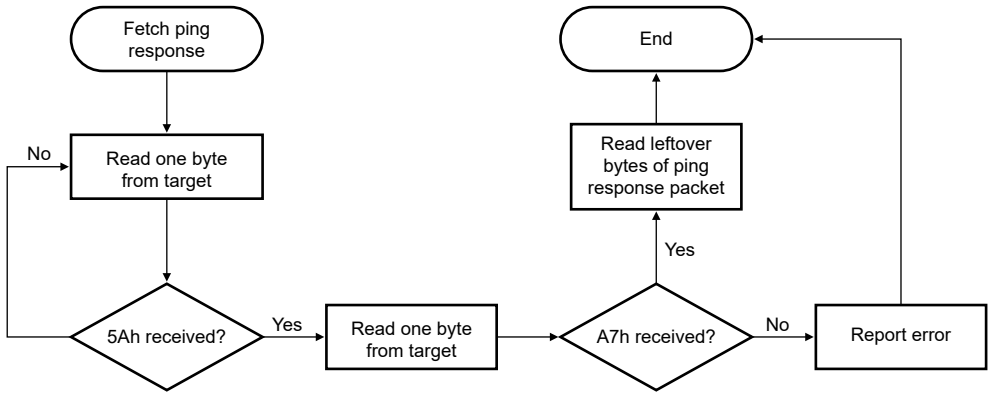


Figure 80. Host reads ping response from target via CAN

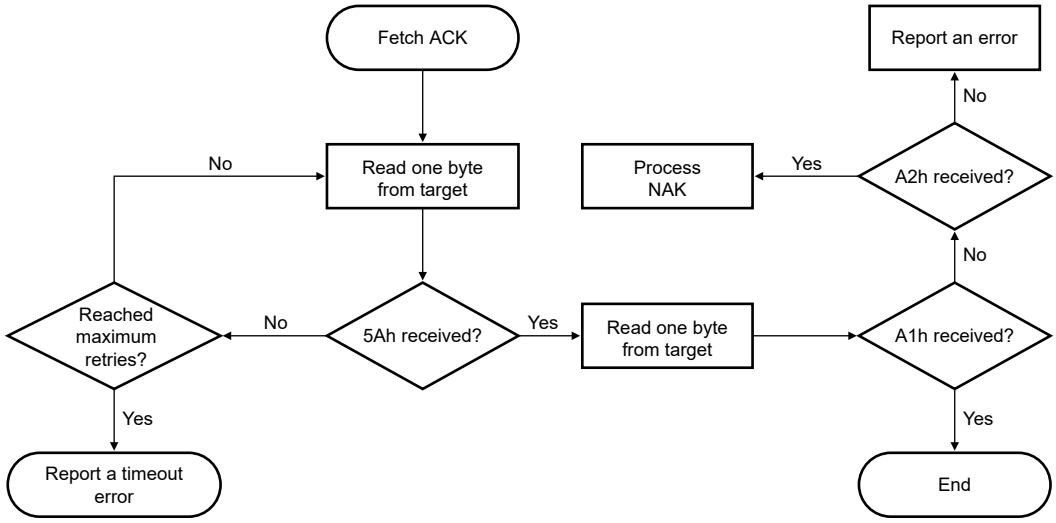


Figure 81. Host reads ACK packet from target via CAN

Chapter 15
Boot ROM

15.1 Overview

This chip supports both on-chip flash memory image boot and external NOR flash memory image boot via the FlexSPI interface.

You can use the boot ROM to erase, program, and read the on-chip or external flash memory, which means you can use the boot ROM to download the boot image into the on-chip or external flash memory via the ISP interfaces.

Following are the boot modes available for the boot ROM:

- Master boot mode
- ISP boot mode
- Recovery boot mode

The boot ROM uses the ISP pins or CMPA configuration to select Master boot mode for on-chip flash memory boot and FlexSPI boot (see Table 226 for more information).

Also, the boot ROM takes responsibility of the boot flow. It selects whether to boot from on-chip flash memory, external flash memory, or ISP mode. It even supports secure boot such as image authentication by ECDSA (P256, P384) or CMAC (128 bits).

There are many boot-related parameters in customer manufacturing/factory programmable area (CMPA) and customer in-field programmable area (CFPA). You can use the boot ROM to update these settings in ISP mode or by using ROM API.

For all details related to CMPA and CFPA, including their fields and address settings, see the IFR map file MCXNP184M150F70_IFR_Fusemap.xlsx, attached to this document.

NOTE

This chapter is also available in the *MCX N Security Reference Manual*, with differences related to security functionality. The chapter in the Reference Manual does not show the security relevant information in it.

15.1.1 Features

The boot ROM (256 KB on-chip with bootloader) allows various boot options and APIs. It supports:

- Automatic booting from internal flash memory, based on ISP pins or the CMPA setting in the PFR.
- In-application programming (IAP) calls (see IAP APIs for more information).
- Flash memory API for programming internal and external FlexSPI NOR flash memory. See the following for more information:
  - FLASH APIs
  - FLEXSPI FLASH Driver APIs
- SPI flash memory recovery boot from 1-bit SPI flash memory devices (see SPI flash memory recovery for more information).

15.2 Functional description

15.2.1 Boot mode and ISP download modes based on the ISP pin

The internal boot ROM memory stores the boot code. After a reset, the Arm processor starts its code execution from this memory. You must execute the bootloader code every time the part is powered on, reset, or wakes up from a deep power-down mode.

Images must be stored in the flash memory or in the serial NOR flash memory connected to the FlexSPI controller. The code is then validated, and the boot ROM vectors to on-chip or off-chip flash memory.

Depending on the values of the CMPA fields, ISP pins, and image header type definition, the bootloader decides whether to boot from flash memory, off-chip flash memory, or run into ISP mode. The boot ROM reads the status of the ISP pins to determine the boot source. See Table 225 for details.

Table 225. Boot mode and ISP download modes based on the ISP pin

Boot mode	ISPMODE_N (P0_6 pin)	Description
Flash memory boot	High	The boot ROM looks for a valid image in the flash memory. If it does not find a valid image, it looks for one in the recovery boot device, if the recovery boot is enabled (9:8, word 0 in CMPA). If it still fails to find a valid image, the boot ROM enters ISP boot mode based on the ISP_BOOT_IF bits defined in Table 226.
ISP boot	Low	One of the serial interfaces (UART, I^2C, SPI, USBFS0, or USBHS1) is used to download the image from the host into the flash memory. The first valid probe message on UART, I^2C, SPI, CAN, or USB becomes inactive on that interface.

NOTE

For the default USB HS ISP mode, you require crystal 24 MHz that you can configure in CMPA. For USB FS ISP mode, the boot ROM uses FRO.

15.2.2 ISP download mode based on the ISP_BOOT_IF CMPA bits

Table 226. ISP download mode based on the ISP_BOOT_IF CMPA bits (word0[6:4])

ISP boot mode	ISP_MODE_2	ISP_MODE_1	ISP_MODE_0	Functionality
Auto ISP	0	0	0	The boot ROM probes the active peripheral from one of the following serial interfaces and downloads the image from the probed peripherals: • UART • I^2C • SPI • CAN • USBFS0 • USBHS1
UART ISP	0	0	1	UART is used to download the image.
SPI ISP	0	1	0	The SPI slave is used to download the image.
I^2C slave ISP	0	1	1	The I^2C slave is used to download the image.
USB0 slave ISP	1	0	0	The USB HID class is used to download the image of the USB0 port.
USB1 slave ISP	1	0	1	The USB HID class is used to download the image of the USB1 port.

Table continues on the next page...

Table 226. ISP download mode based on the ISP_BOOT_IF CMPA bits (word0[6:4]) (continued)

ISP boot mode	ISP_MODE_2	ISP_MODE_1	ISP_MODE_0	Functionality
FlexCAN slave ISP	1	1	0	The FlexCAN slave is used to download the image.
Disable ISP	1	1	1	Disable all peripherals

15.2.3 Boot ROM pin assignments

Table 227 shows ISP pin assignments (the default ones) that the boot ROM code uses. You can change these assignments by using the CMPA settings (see the IFR map file attached to this document for more information).

Table 227. Boot ROM pin assignments

ISP pin	Port pin assignment
ISPMODE_N	P0_6
UART ISP mode	
FC4_UART_TXD	P1_9
FC4_UART_RXD	P1_8
I²C ISP mode	
FC0_I2C_SDA	P0_16
FC0_I2C_SCL	P0_17
SPI Flash Memory Recovery mode	
FC7_SPI_SDO	P3_9
FC7_SPI_SDI	P3_8
FC7_SPI_PCS0	P3_0
FC7_SPI_SCK	P3_7
SPI ISP mode	
FC3_SPI_SCK	P1_1
FC3_SPI_PCS	P1_3
FC3_SPI_SDO	P1_0
FC3_SPI_SDI	P1_2
USBFS0 and USBHS1 ISP mode	
USB0_VBUS	P4_12

Table continues on the next page...

Table 227. Boot ROM pin assignments (continued)

ISP pin	Port pin assignment
USB0_DP	—
USB0_DM	—
USB1_VBUS	—
USB1_DP	—
USB1_DM	—
FlexCAN ISP mode	
CAN_TXD	P1_10
CAN_RXD	P1_11
FlexSPI pins	
FLEXSPI0_A_SS0_b	P3_0
FLEXSPI0_A_SCLK	P3_7
FLEXSPI0_A_DQS	P3_6
FLEXSPI0_A_DATA0	P3_8
FLEXSPI0_A_DATA1	P3_9
FLEXSPI0_A_DATA2	P3_10
FLEXSPI0_A_DATA3	P3_11
FLEXSPI0_A_DATA4	P3_12
FLEXSPI0_A_DATA5	P3_13
FLEXSPI0_A_DATA6	P3_14
FLEXSPI0_A_DATA7	P3_15
FLEXSPI0_B_SS0_b	P2_2
FLEXSPI0_B_SCLK	P2_3
FLEXSPI0_B_DQS	P2_1
FLEXSPI0_B_DATA0	P2_4
FLEXSPI0_B_DATA1	P2_5

Table continues on the next page...

Table 227. Boot ROM pin assignments (continued)

ISP pin	Port pin assignment
FLEXSPI0_B_DATA2	P2_6
FLEXSPI0_B_DATA3	P2_7
FLEXSPI0_B_DATA4	P2_8
FLEXSPI0_B_DATA5	P2_9
FLEXSPI0_B_DATA6	P2_10
FLEXSPI0_B_DATA7	P2_11
FLEXSPI_RESET	See the IFR map file attached to this document.

15.2.4 Image header

Top-level boot flow shows the top-level boot process, which starts after the reset is released.

The CPU clock has a default frequency of 48 MHz, based on 144 MHz FRO by default. You can set this frequency to 72 MHz, 144 MHz, or 150 MHz by using the BOOT_SPEED setting in CMPA. When the Cortex-M33 core starts the bootloader, the SWD access is disabled; and therefore, the debugger cannot connect to the CPU during this period. The boot ROM determines the boot mode based on the reset state of the ISP pins.

After completing boot mode detection, the bootloader starts validating the vector table and image header if the image is present in the boot media (internal or external flash memory).

The boot ROM does the following for image validity check:

- Validates the image using header and CMPA settings when secure boot is enabled. See the "Secure Boot ROM" chapter in the *Security Reference Manual* for details.
- Validates the image using CRC32 when secure boot is not enabled, if the CRC check is present in the image header.
- Validates whether SP and PC are valid RAM and NVM addresses of the chip.
- Validates the TZ-M image type if the aforementioned image checks pass.

The beginning of the image follows the format described in Table 228. The bootloader begins scanning for user images by examining the image type marker located at offset 24h. If the value matches any supported image type markers, the image header's validation begins, and after its completion, the qualification continues by examining the TZM "imageType" field:

- If it is a CRC image, you use the "image length" field value as the length to perform a CRC calculation based on the image in the flash memory (see Table 228 for more). The CRC calculation begins at offset 0h from the beginning of the image sector and continues up to the number of bytes specified by the image length, which does not include the "offsetToExtendedHeader" field that makes up the CRC value field. This means that the calculated CRC skips the CRC value field. The result is then compared to the "offsetToExtendedHeader" entry in the structure and the image is considered valid if a match exists; otherwise, the image is considered invalid. The CRC calculation is not performed if the image is not a CRC image.
- If it is a signed image, the "image length" field value is used as the length to perform authentication (ECDSA and CMAC), which is performed on the image in the internal or external flash memory. The authentication begins at offset 0h from the beginning of the image sector, and continues up to the number of bytes specified by the image length. The "offsetToExtendedHeader" field value points to the offset that holds the certificates.

Table 228. Image header

Offset (hex)	Size (bytes)	Symbol	Description
00	4	Initial SP	Stack pointer
04	4	Initial PC	Application's first execution instruction
08	24	Vector table	Cortex-M33 core's vector table entries
20	4	image length	Length of the current image (total length, including the signature), which is set to the actual image length if the image type is another value
24	4	imageType	Image type, bit[7:0]: • 0h: Plain image • 2h: Plain image with CRC • 4h: Signed image • 5h: Plain image with CRC • 6h: SB4 manifest Load_to_RAM image: The image has a nonzero image_load_address and image_length in the image header. Other values are reserved. Bit[10]: Indicates whether the image version is included in ImageType[31:16]: • 0: Image version is not included in "imageType." • 1: Image version is included in "Image Type," applicable to the internal flash memory XIP use case only. Bit[13]: Indicates whether the image includes TZM preset data: • 0: Image does not contain TZM preset data. • 1: Image contains TZM preset data. Bits[31:16]: Image version (for on-chip flash memory) when bit[10] = 1
28	4	offsetToExtendedHeader	Offset to extended header: • For a signed image (imageType = 4h), this is the offset to the certificate header block • For a CRC image (imageType = 2h or 5h), this is the crcChecksum value
2C	8	Vector table	Cortex-M33 core's vector table entries
34	4	imageExecutionAddress	The execution address of the image is: • Set to 0 if image type is XIP

Table continues on the next page...

Table 228. Image header (continued)

Offset (hex)	Size (bytes)	Symbol	Description
			• Set to actual image execution address if the image type is loaded to RAM
38	—	Vector table	Cortex-M33 core's vector table entries

15.2.5 Top-level boot flow

The following figure shows the top-level boot process, which begins after the reset is released.

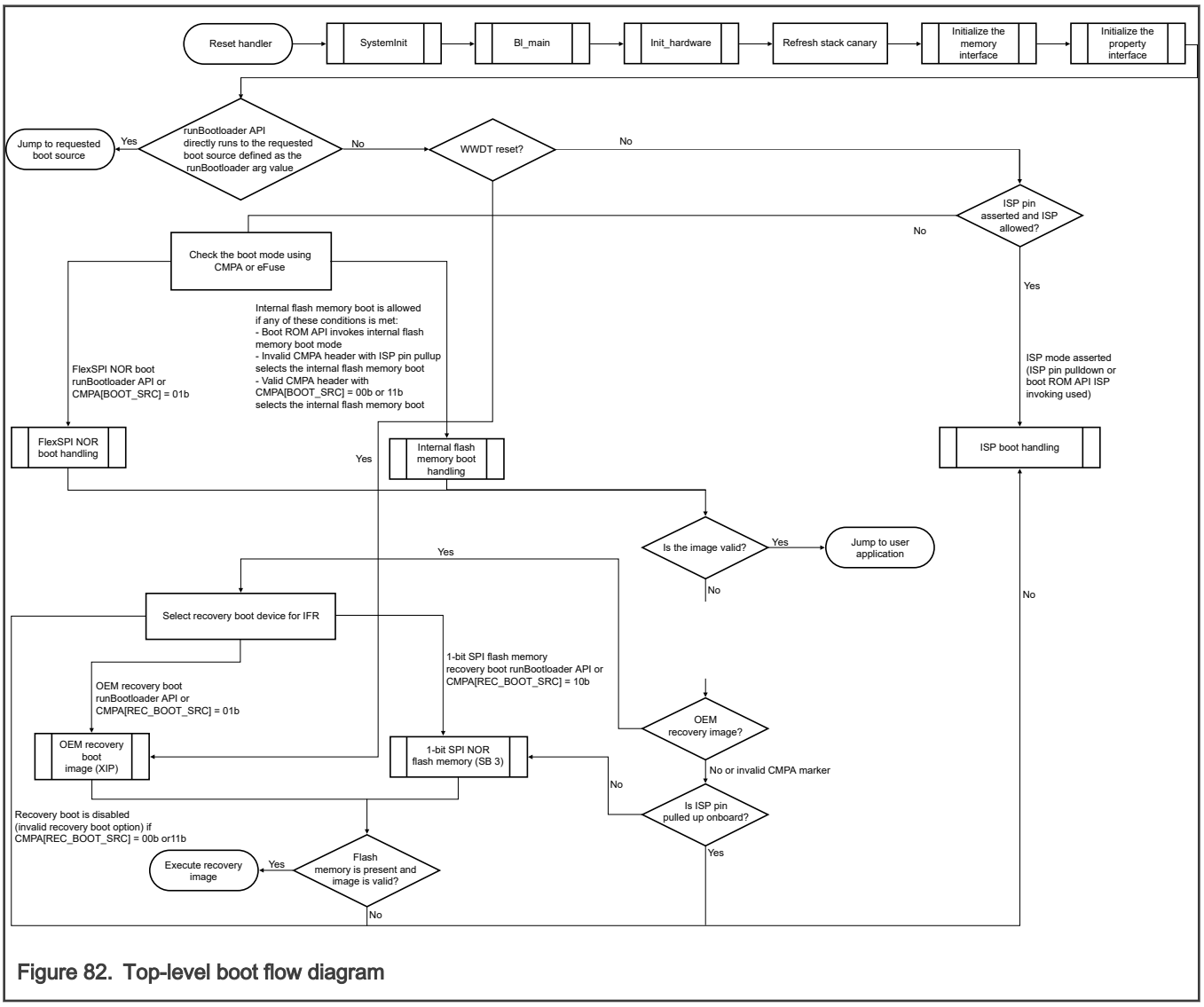


Figure 82. Top-level boot flow diagram

15.2.6 SPI flash memory recovery

Support is provided for a recovery boot from an external 1-bit SPI flash memory device in which the SB3.1 image is stored. The SB3.1 file is an encrypted and signed command script file that supports programming flash memory, IFR, and other configuration commands. You can implement this feature during OTA using a recovery media model in which an external SPI flash memory is

used to store a factory image in the SB3.1 format. If the image in the main flash memory is corrupted, the boot ROM attempts to recover the chip by booting or executing the SB3.1 file that is present on the external flash memory device.

See the "ROM firmware update using SB file" section in the "Secure Boot ROM" chapter in the MCX N Security Reference Manual for details related to the SB3.1 file format.

When booting from an internal or external flash memory using FlexSPI, if the flash memory image is deemed invalid, the chip checks REC_BOOT_SRC (bits 9:8) in the CMPA BOOT_CFG field (0100_4000h) to determine whether SPI flash memory recovery is selected.

The SB file can program the image into internal flash memory. The following commands are available for SB file recovery mode:

```
#define SBLOADER_V3_CMD_SET_IN_REC_MODE \\
    ((1u << kSB3_CmdErase) | (1u << kSB3_CmdLoad) | (1u << kSB3_CmdExecute) |
 (1u << kSB3_CmdProgramFuse) |      \\
    (1u << kSB3_CmdProgramIFR) | (1u << kSB3_CmdCopy) | (1u << kSB3_CmdLoadKeyBlob) |
 (1u << kSB3_CmdConfigMem) | \\
    (1u << kSB3_CmdFillMem))| (1u << kSB3_CmdFwVerCheck))
```

Code Listing 1. Commands for SB file recovery mode

See 1-bit serial NOR flash memory through SPI for more information.

15.3 Boot modes

15.3.1 Master boot mode

Master boot mode supports the following boot devices:

- Internal flash memory boot
- FlexSPI NOR flash memory boot
- SPI 1-bit NOR recovery boot
- Secondary bootloader boot (SBL)
- Bank1_IFR0 recovery boot (OEM_REC)

Table 229. Image offsets for different boot media

Boot media	Image offset
Internal flash memory boot	0h
FlexSPI NOR flash memory boot	1000h
SPI 1-bit NOR recovery boot	0h
Secondary bootloader boot	0h
Bank1_IFR0 recovery boot	0h

15.3.1.1 Internal flash memory boot

The CPU clock is set to the boot speed (that the BOOT_SPEED field specifies) in the CMPA region and boots directly from the internal flash memory or external FlexSPI NOR flash memory based on the BOOT_SRC settings in CMPA. After you select the internal flash memory boot as the boot media, the boot ROM tries to find a valid boot image in the internal flash memory. For more information, see Boot flow.

15.3.1.1.1 Boot flow

Figure 83 shows internal flash memory boot flow:

- The CMPA[FLASH_REMAP_SIZE] field specifies the second image in terms of size. If the field is 0, it indicates that the second image is not present.
- The Secure_FW_Version field [31:0], word 2 in CFPA specifies the minimum allowed image version.
- If the image version flag at imageType[10] is set, the image version can be found at imageType[31:16]; otherwise, the image version is treated as 0.
- In the image list, the newer one is image 0, and the older one is image 1. If the image version information is missing or equal, the image starting from address 0 is treated as image 0, and the image that FLASH_REMAP_SIZE specifies is treated as image 1, which starts from the second bank of the flash memory when the flash memory swap is enabled.

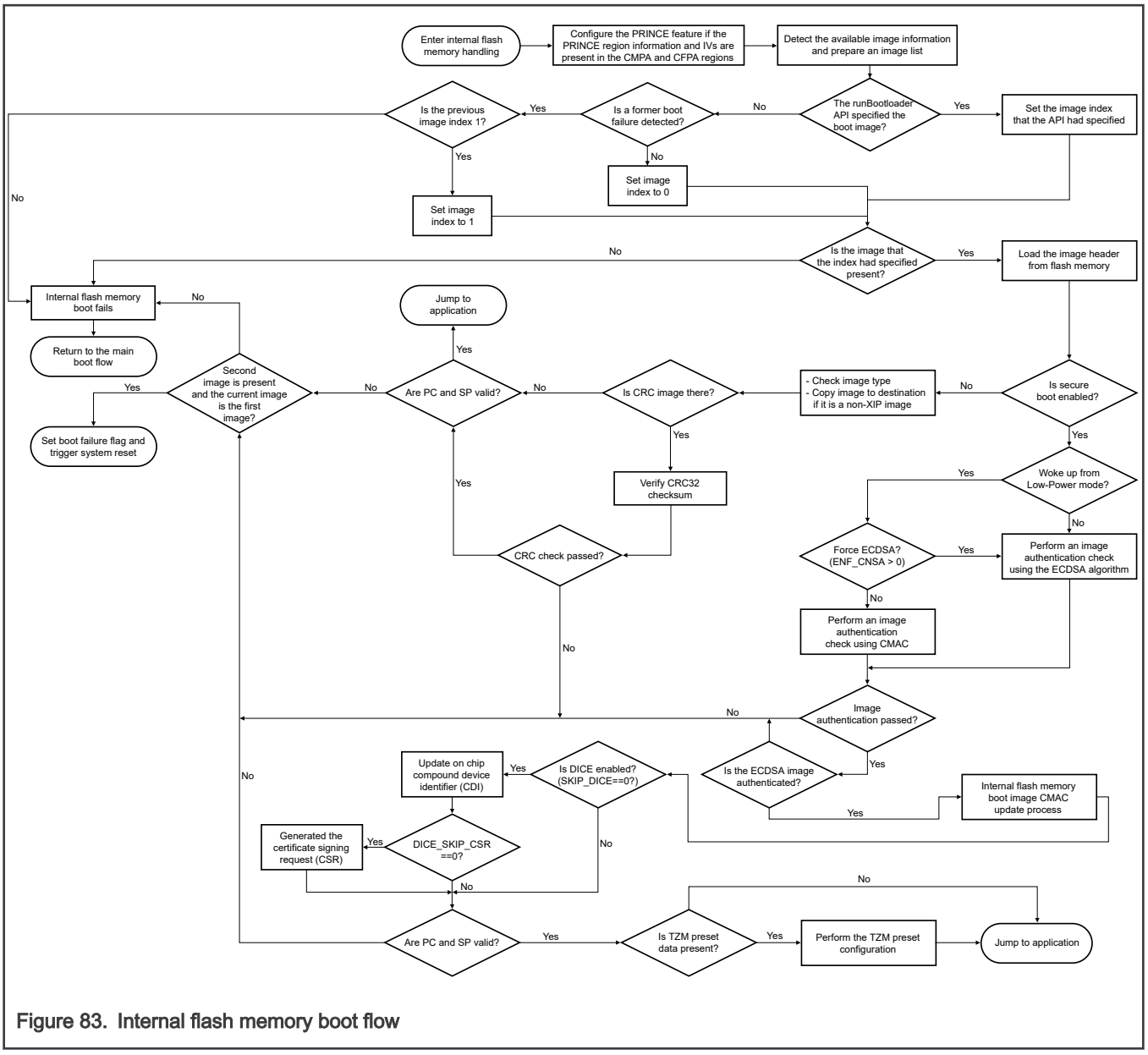


Figure 83. Internal flash memory boot flow

15.3.1.1.2 Internal flash memory dual image boot

The boot ROM supports dual image boot for internal flash memory. This means that two boot images can be placed in the flash memory region. The boot ROM decides which image to boot first based on the image version. It boots the one with the newer image version first, and in case of a failure, it boots the older one. The image version is the image header offset, 24h (see Table 228). Bit 10 shows whether the image contains the image version; if bit 10 is 0, it means that the image has no image version. In this case, the boot ROM considers the image version as 0.

Table 230. Internal flash memory boot image version (image header offset: 24h)

image_type	Field name	Field description
Bits 31–16	Image version	Active when the remap feature is enabled (remap offset and remap size are not zero in CMPA)
Bit 10	IMG_VER_INCLUDED_IN_IMG_TYPE	If this field is 0, the image has no version. If this field is 1, the image has a version in the dual image boot.

15.3.1.1.2.1 Remap feature

The following figure shows the internal flash memory controller remap function. When FLASH_REMAP_SIZE is a nonzero value in CMPA, the internal flash memory AHB access changes the access address based on the current active flash memory bank number. For example, when FLASH_REMAP_SIZE = 2h, and the remap size is 96 KB ((2h + 1) × 32 KB), access to 0h is remapped to flash memory bank1 start offset 0 when the bank1 image is active. The boot ROM can implement a dual image boot with two images via this feature. You set the remap size of the image in the CMPA and eFuse regions.

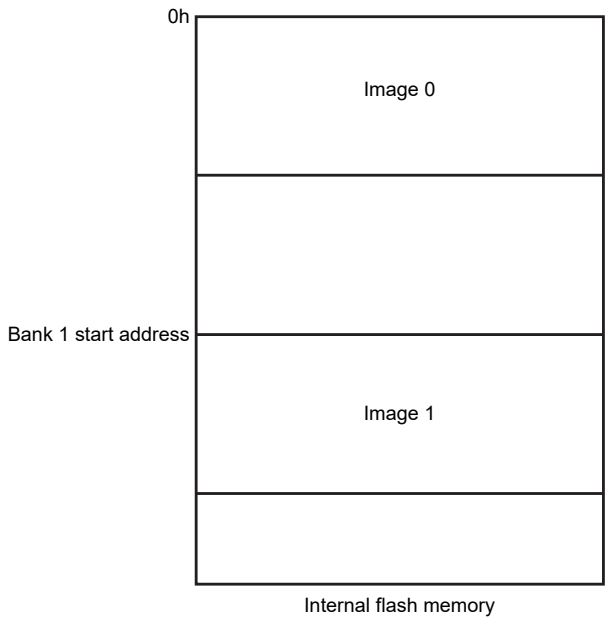


Figure 84. Internal flash memory dual image layout

15.3.1.1.2.2 Remap settings

The following table shows the internal flash memory dual image boot remap setting in CMPA. You must perform the settings defined in this table in CMPA to let the boot ROM know where the second image is placed. You can, then, enable dual image boot by setting the dual image boot image version in the image header, according to the information provided in the aforementioned sections. For more information about the internal flash memory boot flow, see Boot flow.

Table 231. Boot image 1 remap size

Region	Address	Word name	Description
CMPA	0100_4004h	FLASH_REMAP_SIZE	Indicates the flash memory remap size from the flash memory bank1 address, 0h, to the boot ROM.
eFuse	Index: 25	FLASH_REMAP_SIZE[4:0]	Indicates the flash memory remap size from the flash memory bank1 address, 0h, to the boot ROM. Also, nonzero FLASH_REMAP_SIZE[4:0] in eFuse replaces the FLASH_REMAP_SIZE[4:0] data in the CMPA field.

15.3.1.1.2.3 XOM access control settings

In some scenarios, you may need to set some internal flash memory regions as execute-only-memory (XOM) or read-only memory (ROM), or hide flash memory regions to protect the code.

For example, if ACL_SEC_0 = 2, it means that the first sector (32 KB) of the flash memory global access control (GLBAC*n*) index is 2. As shown in Table 232, this sector is set as ROM.

Table 232. Flash memory MBC setting index

GLBAC*n* index	Read	Write	Execute	Lock	Description
0	1	1	1	0	Default flash memory behavior (R/W/X unlocked)
1	1	1	0	1	Data flash memory with this setting: R/W + locked
2	1	0	1	1	ROM with this setting: RX + locked
3	1	0	0	1	Data read-only memory (DROM) with this setting: R + locked
4	1	0	1	0	ROM with this setting: RX unlocked
5	0	0	1	0	XOM with this setting: XOM unlocked
6	0	0	1	1	XOM with this setting: XOM + locked
7	0	0	0	1	Hidden (no access + locked)

Follow below rules to change the ACL_SEC_*n* fields.

- After you select a locked access level, the subsequent updates of this field can be done with higher lock level only.
  - If current sector ACL value (GLBAC*n* index) is greater than the new value, then it is permitted except if the current value is 4 or 5.
    - 7 (___L) > 6 (__XL) > 3 (R__L) > 2 (R_XL) > 1 (RW_L) > 0 (RWX_)
  - If current sector ACL value is 0, 4, or 5 then any new value is permitted.

The boot ROM supports setting flash memory attributes in the CFPA region. Before jumping into the user code, the boot ROM loads the settings of access control for the flash memory sector that you specify in CFPA. Table 233 shows access control settings for flash memory in CFPA.

Table 233. Access control settings for the flash memory sector

CFPA word	Flash memory address (hex)	Word name	Description
Word48	0100_00C0	FLASH_ACL_0_7	Selects the flash memory MBC setting index. See Table 232 for more information.
Word49	0100_00C4	FLASH_ACL_8_15	Same as above
Word50	0100_00C8	FLASH_ACL_16_23	Same as above
Word51	0100_00CC	FLASH_ACL_24_31	Same as above
Word52	0100_00E0	FLASH_ACL_32_39	Same as above
Word53	0100_00E4	FLASH_ACL_40_47	Same as above
Word54	0100_00E8	FLASH_ACL_48_55	Same as above
Word55	0100_00EC	FLASH_ACL_55_63	Same as above

15.3.1.1.2.4 CMPA settings with PRINCE enabled

The following figure shows CMPA settings for internal flash dual image boot with PRINCE enabled.

Table 234. Remapping settings for dual image

CMPA word	Flash memory address	Word name	Value	Details
WORD1	0100_4004h	FLASH_REMAP_SIZE	1h	Remap size: 10000h

After all settings are updated, the boot ROM follows dual image boot as the boot flow progresses.

15.3.1.2 FlexSPI NOR flash memory boot

The boot ROM supports access to different Quad and Octal SPI NOR flash memory chips from various vendors via the FlexSPI interface using 1-bit, 4-bit (Quad), or 8-bit (Octal) mode. It uses the flash memory configuration block (FCB) located at offset 400h on the flash memory chip or flash memory auto-probe feature that CMPA specifies:

- If FLEXSPI_AUTO_PROBE_EN is blown, the bootloader performs the flash memory auto-probe sequence using parameters blown in:
  - The FLEXSPI_PROBE_TYPE field that defines the flash memory auto-probe type.
  - The FLEXSPI_PROBE_TYPE and FLEXSPI_FLASH_TYPE fields that define the flash memory type.
  - The FLEXSPI_FREQUENCY field that defines the flash memory access speed.

  The auto-probe feature is used for detecting the external flash FCB information automatically. When FLEXSPI_AUTO_PROBE_EN is blown, and the preceding necessary fields are programmed, the boot ROM tries to identify the external flash information. If the identification is successful, it obtains a self-identifying FCB configuration block. If the recognition fails, it continues to attempt to read the user FCB configuration block to obtain external flash information.

  When FLEXSPI_AUTO_PROBE_EN=1, which is only supported on chips that are JESD216-compliant, the boot ROM is able to detect the flash memory information via the read SFDP (5Ah) command. The ROM uses the SFDP information to construct an FCB instead of using one from the application image.

- If FLEXSPI_AUTO_PROBE_EN is not blown, the bootloader looks at offset 400h on the flash memory chip; if data at offset 400h is equal to 4246_4346h, the bootloader reads the whole 512-byte FCB into the on-chip SRAM and configures the FlexSPI controller using this FCB accordingly.

  After performing the aforementioned operation, the bootloader starts performing the normal boot flow. See Figure 85 for more information.

Table 235. FlexSPI FCB settings

Field	Offset (hex)	Size (bytes)	Description
tag	000	4	Configuration block tag Setting: 4246_4346h
version	004	4	Configuration block that the boot ROM uses internally Setting: 5601_0400h
Reserved	008	4	Reserved for future use
readSampleClkSrc	00C	1	Read sampling clock source options: 0 – Internal sampling 1 – Loopback from DQS pad 2 – Loopback from SCK pad 3 – External DQS signal
csHoldTime	00D	1	CS hold time in terms of flash memory clock cycles Recommended value is 3.
csSetupTIme	00E	1	CS set up time in terms of flash memory clock cycles The recommended value is 3.
columAddressWidth	00F	1	Column address width: • Is set to 3 for HyperFlash NOR flash memories. • Is set to 0 for other flash memory chips.
deviceModeCfgEnable	010	1	Enables the chip mode configuration sequence: 1 – Enable 0 – Disable
deviceModeType	011	1	Argument for Device mode configuration For example, Quad enable setting in Status Register2 for some Quad SPI flash memory devices: 0 – No mode change 1 – Quad enable (switch from SPI to Quad mode) 2 – Spi2Xpi (switch from SPI to DPI, QPI, or OPI mode) 3 – Xpi2Spi (switch from DPI, QPI, or OPI to SPI mode)

Table continues on the next page...

Table 235. FlexSPI FCB settings (continued)

Field	Offset (hex)	Size (bytes)	Description
waitTimeCfgCommands	012	2	Wait time (in terms of 100 µs) for Device mode configuration command
deviceModeSeq	014	4	Device mode configuration sequence: Byte 0 – Number of required sequences Byte 1 – Sequence index
deviceModeArg	018	4	Argument for Device mode configuration For example, Quad enable setting in Status Register2 for some Quad SPI flash memory devices
configCmdEnable	01C	1	Configuration command enable: 0 – Ignores configuration command 1 – Enables configuration command
configModeType	01D	3	Configuration command type that supports up to three types; each type is combined with a configuration command sequence
configCmdSeqs	020	12	Configuration command that supports up to three sequences
Reserved	02C	4	Reserved for future use
configCmdArgs	030	12	Configuration command that supports up to three arguments
Reserved	03C	4	Reserved for future use
controllerMiscOption	040	4	Miscellaneous controller configuration options: Bit 0 – Differential clock enable: set to 1 for HyperFlash NOR flash memory 1V8 device and set to 0 for other devices Bit 3 – WordAddressableEnable: set to 1 for HyperFlash NOR flash memory and set to 0 for other devices Bit 4 – SafeConfigFreqEnable: set to 1 if expecting to configure the chip with a safe frequency Bit 6 – DDR mode enable: set to 1 if DDR read is expected Other bits – Reserved; set to 0
deviceType	044	1	Device type 1 – Serial NOR flash memory
sflashPadType	045	1	Data pad used in Read command: 1 – Single pad 2 – Dual pads 4 – Quad pads

Table continues on the next page...

Table 235. FlexSPI FCB settings (continued)

Field	Offset (hex)	Size (bytes)	Description
			8 – Octal pads
serialClkFreq	046	1	Serial clock frequency SDR: 1 – 30MHz 2 – 50MHz 3 – 60MHz 4 – 75MHz 5 – 100MHz DDR: 1 – 30MHz 2 – 50MHz
lutCustomSeqEnable	047	1	LUT customization enable—required when you cannot perform the program and erase operations using one LUT sequence (currently, only one is applicable to HyperFlash NOR flash memory)
Reserved	048	8	Reserved for future use
sflashA1Size	050	4	Size of flash memory connected to A1
sflashA2Size	054	4	Size of flash memory connected to A2
sflashB1Size	058	4	Size of flash memory connected to B1
sflashB2Size	05C	4	Size of flash memory connected to B2
csPadSettingOverride	060	4	Pad override value for CS pin Use this value (if nonzero) to configure the CS pin; otherwise, use the default boot ROM setting
sclkPadSettingOverride	064	4	Serial clock pad setting override value
dataPadSettingOverride	068	4	Data pad setting override value
dqsPadSettingOverride	06C	4	DQS pad setting override value
timeoutInMs	070	4	Timeout value (in terms of ms) to terminate the busy check
commandInterval	074	4	CS deselect interval between two commands
dataValidTime	078	4	Clock edge to data valid time
busyOffset	07C	2	Busy offset, valid value: 0-31

Table continues on the next page...

Table 235. FlexSPI FCB settings (continued)

Field	Offset (hex)	Size (bytes)	Description
busyBitPolarity	07E	2	Busy flag polarity: 0 – Busy flag is 1 when the flash memory device is busy 1 – Busy flag is 0 when the flash memory device is busy
lookupTable	080	256	16 LUT sequences—each sequence consists of four words (see the FlexSPI chapter for more information)
lutCustomSeq	180	48	Customizable LUT sequences
Reserved	1B0	16	Reserved for future use
pageSize	1C0	4	Page size of flash memory
sectorSize	1C4	4	Sector size of flash memory
ipcmdSerialClkFreq	1C8	1	Serial clock frequency for IP commands: 0 – The same with read command Others – The same definition as that of the serial clock frequency
isUniformBlockSize	1C9	1	Indicates whether the sector size is the same as the block size: 0 – No 1 – Yes
isDataOrderSwapped	1CA	1	Data order (D0, D1, D2, D3) is swapped with (D1, D0, D3, D2)
Reserved	1CB	1	Reserved for future use
serialNorType	1CC	1	Serial NOR flash memory type: 0, 1, 2, and 3
needExitNoCmdMode	1CD	1	Exits NoCmd mode before other IP commands
halfClkForNonReadCmd	1CE	1	Half serial clock for nonread command: true or false
needRestoreNoCmdMode	1CF	1	Restores NoCmd mode after IP command execution
blockSize	1D0	4	Flash memory block size
flashStateCtx	1D4	4	Flash memory state context
Reserved	1D8	40	Reserved for future use

15.3.1.2.1 Reset

During FlexSPI boot, the boot process requires the FlexSPI flash memory device to be in a certain mode, for example, 1-bit SPI compatible mode. The flash memory device is, by default, in this mode after a POR reset. That is because the power-up sequence resets it along with the boot ROM device. However, the flash memory device does not remain in 1-bit SPI compatible mode if you configure it in DPI, QPI, or Octal mode when any non-POR resets happen (for example, watchdog timer and external pin reset).

In such a case, the boot process requires special processing to restore the flash memory device to 1-bit SPI compatible mode before accessing it. You can achieve this by using a GPIO to assert a reset pin on the flash memory device. The bootloader can perform the reset process and reset the flash memory device to 1-bit SPI compatible mode if the FLEXSPI_RESET_ENABLE field is a nonzero value in CMPA, using the GPIO that the FLEXSPI_RESET_GPIO_PORT and FLEXSPI_RESET_GPIO_PIN fields specify in combination.

NOTE

For Octal flash memory, after the boot ROM boots up, the flash memory is configured to Octal mode. The boot ROM cannot reset the NOR flash memory from Octal mode to standard SPI mode if it is a pin or soft reset causing reboot fail. The only way to solve this is to use a GPIO to reset the Octal NOR flash memory during reboot. There is no such problem in case of Quad SPI flash memory.

Table 236. FlexSPI boot configurations in CMPA WORD32(0100_4010h)(FLEXSPI_BOOT_CFG[0])

Field name	Enum name	Description	Offset	Width	Value (binary)
FLEXSPI_AUTO_PROBE_EN	—	Quad SPI or Octal SPI flash memory auto-probe feature enable	0	1	—
FLEXSPI_PROBE_TYPE	—	Quad SPI or Octal SPI flash memory probe type	1	3	—
FLEXSPI_PROBE_TYPE	QUADSPI_NOR	Quad SPI NOR flash memory	—	—	000b
FLEXSPI_PROBE_TYPE	MICRON_OCTAL	Micron Octal flash memory	—	—	001b
FLEXSPI_PROBE_TYPE	MACRONIX_OCTAL	Macronix Octal flash memory	—	—	010b
FLEXSPI_PROBE_TYPE	ADESTO_OCTAL	Adesto Octal flash memory	—	—	011b
FLEXSPI_PROBE_TYPE	—	Reserved	—	—	100b
FLEXSPI_PROBE_TYPE	—	Reserved	—	—	101b
FLEXSPI_PROBE_TYPE	—	Reserved	—	—	110b
FLEXSPI_PROBE_TYPE	—	Reserved	—	—	111b
FLEXSPI_FLASH_TYPE	—	Typical serial NOR flash memory types	4	3	—
FLEXSPI_FLASH_TYPE	QSPI_ADDR_3B	Quad SPI flash memory address (by default, the chip supports 3B read)	—	—	000b
FLEXSPI_FLASH_TYPE	—	Reserved	—	—	001b
FLEXSPI_FLASH_TYPE	—	Reserved	—	—	
FLEXSPI_FLASH_TYPE	HYPER_3V3	HyperFlash NOR flash memory 3V3	—	—	011b

Table continues on the next page...

Table 236. FlexSPI boot configurations in CMPA WORD32(0100_4010h)(FLEXSPI_BOOT_CFG[0]) (continued)

Field name	Enum name	Description	Offset	Width	Value (binary)
FLEXSPI_FLASH_TYPE	OSPI_DDR_MXIC	MXIC Octal DDR	—	—	100b
FLEXSPI_FLASH_TYPE	OSPI_DDR_MICRON	Micron Octal DDR	—	—	101b
FLEXSPI_FLASH_TYPE	—	Reserved	—	—	110b
FLEXSPI_FLASH_TYPE	—	Reserved	—	—	111b
FLEXSPI_DUMMY_CYCLES	—	Quad SPI or Octal SPI dummy cycles for read command	7	4	—
FLEXSPI_DUMMY_CYCLES	PROBE_DUMMY_CYCLE	Dummy cycles probed automatically	—	—	0000b
FLEXSPI_DUMMY_CYCLES	—	Number of dummy cycles	—	—	Other value
FLEXSPI_FREQUENCY	—	Quad SPI or Octal SPI flash interface frequency	11	3	—
FLEXSPI_FREQUENCY	FLEXSPI_75MHZ	75 MHz	—	—	000b
FLEXSPI_FREQUENCY	FLEXSPI_60MHZ	60 MHz	—	—	001b
FLEXSPI_FREQUENCY	FLEXSPI_50MHZ	50 MHz	—	—	010b
FLEXSPI_FREQUENCY	FLEXSPI_100MHZ	100 MHz	—	—	011b
FLEXSPI_FREQUENCY	—	Reserved	—	—	—
FLEXSPI_FREQUENCY	—	Reserved	—	—	—
FLEXSPI_FREQUENCY	—	Reserved	—	—	—
FLEXSPI_FREQUENCY	—	Reserved	—	—	—
FLEXSPI_RESET_PIN_ENABLE	—	FLEXSPI_RESET_PIN to use to reset the flash memory device	14	1	—
FLEXSPI_RESET_PIN_ENABLE	NO_RESET	Quad SPI or Octal SPI flash memory device reset pin is not connected or is unavailable	—	—	—
FLEXSPI_RESET_PIN_ENABLE	EN_RESET	Quad SPI or Octal SPI flash memory device reset pin is connected to a GPIO (FLEXSPI_RESET_PIN)	—	—	—

Table continues on the next page...

Table 236. FlexSPI boot configurations in CMPA WORD32(0100_4010h)(FLEXSPI_BOOT_CFG[0]) (continued)

Field name	Enum name	Description	Offset	Width	Value (binary)
FLEXSPI_RESET_PIN	—	GPIO port and pin number to use for the Quad SPI or Octal SPI flash memory reset function	15	8	—
FLEXSPI_RESET_PIN	GPIO_PORT	GPIO port	0	3	—
FLEXSPI_RESET_PIN	GPIO_PIN_NUM	GPIO pin number	3	5	—
FLEXSPI_HOLD TIME	—	Wait time before accessing serial flash memory	23	2	—
FLEXSPI_HOLD TIME	WAIT_500_US	500 µs	—	—	00b
FLEXSPI_HOLD TIME	WAIT_1_MS	1 ms	—	—	01b
FLEXSPI_HOLD TIME	WAIT_3_MS	3 ms	—	—	10b
FLEXSPI_HOLD TIME	WAIT_10_MS	10 ms	—	—	11b
FLEXSPI_PWR_HOLD_TIME	—	Delay (wait time) after POR before accessing Quad SPI or Octal SPI flash memory devices, in addition to the delay that the QSPI_HOLD TIME field specifies	25	4	—
FLEXSPI_PWR_HOLD_TIME	NO_DELAY	No delay	—	—	0000b
FLEXSPI_PWR_HOLD_TIME	100US_DELAY	Additional 100 µs	—	—	0001b
FLEXSPI_PWR_HOLD_TIME	500US_DELAY	Additional 500 µs	—	—	0010b
FLEXSPI_PWR_HOLD_TIME	1MS_DELAY	Additional 1 ms	—	—	0011b
FLEXSPI_PWR_HOLD_TIME	10MS_DELAY	Additional 10 ms	—	—	0100b
FLEXSPI_PWR_HOLD_TIME	20MS_DELAY	Additional 20 ms	—	—	0101b
FLEXSPI_PWR_HOLD_TIME	40MS_DELAY	Additional 40 ms	—	—	0110b
FLEXSPI_PWR_HOLD_TIME	60MS_DELAY	Additional 60 ms	—	—	0111b
FLEXSPI_PWR_HOLD_TIME	80MS_DELAY	Additional 80 ms	—	—	1000b
FLEXSPI_PWR_HOLD_TIME	100MS_DELAY	Additional 100 ms	—	—	1001b
FLEXSPI_PWR_HOLD_TIME	120MS_DELAY	Additional 120 ms	—	—	1010b
FLEXSPI_PWR_HOLD_TIME	140MS_DELAY	Additional 140 ms	—	—	1011b

Table continues on the next page...

Table 236. FlexSPI boot configurations in CMPA WORD32(0100_4010h)(FLEXSPI_BOOT_CFG[0]) (continued)

Field name	Enum name	Description	Offset	Width	Value (binary)
FLEXSPI_PWR_HOLD_TIME	160MS_DELAY	Additional 160 ms	—	—	1100b
FLEXSPI_PWR_HOLD_TIME	180MS_DELAY	Additional 180 ms	—	—	1101b
FLEXSPI_PWR_HOLD_TIME	200MS_DELAY	Additional 200 ms	—	—	1110b
FLEXSPI_PWR_HOLD_TIME	220MS_DELAY	Additional 220 ms	—	—	1111b
PORT_SEL	—	—	31	1	—

15.3.1.2.2 Boot flow

Figure 85 illustrates the FlexSPI NOR flash memory boot flow. It shows details related to normal image load and authentication flow:

- The FLEXSPI_BOOT_CFG[0] and FLEXSPI_BOOT_CFG[1] fields in CMPA specify the second image in terms of offset and size. If both are 0, it means the second image is not present.
- If the image version at each flash memory offset 600h is valid, use the image version; otherwise, the image version is treated as 0.
- In the image list, the newer one is image 0, and the older one is image 1. If the image version information is missing or equal, the image starting from address 0 is treated as image 0, and the image that the FLEXSPI_BOOT_CFG[0] and FLEXSPI_BOOT_CFG[1] fields specify is treated as image 1.

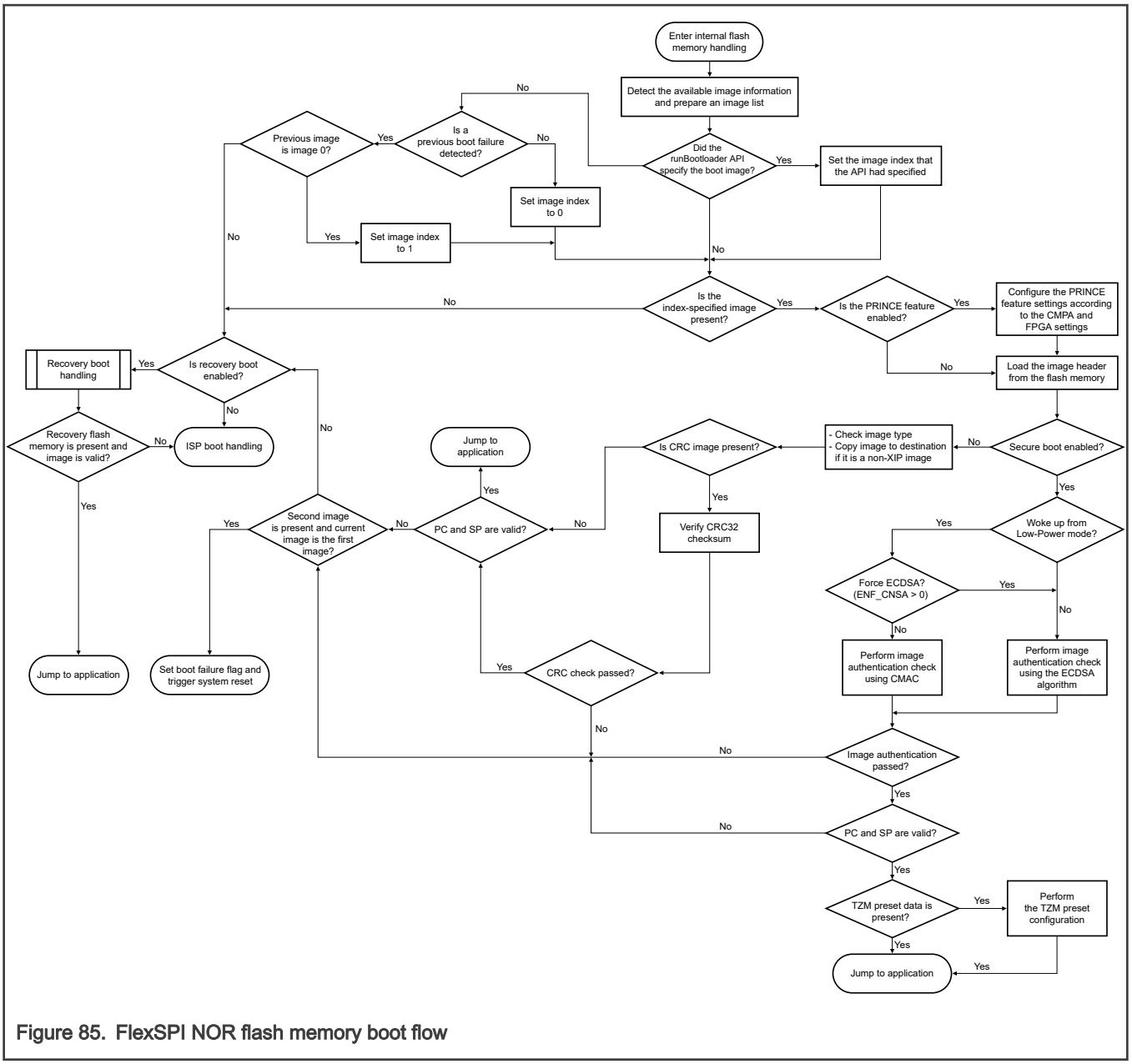


Figure 85. FlexSPI NOR flash memory boot flow

15.3.1.2.3 Dual image ping-pong boot

The FlexSPI controller supports the remap function, which can remap the AHB accessing the address of the serial NOR flash memory to another address with the same offset. In this way, FlexSPI boot supports the second image boot when the first image boots fails.

For dual image ping-pong boot, each of the boot images has its image version, which the boot ROM uses to first select the latest boot image. If the latest image boot fails, the boot ROM uses the old image to boot again.

15.3.1.2.4 Basic remapping function

The following figure shows the basic FlexSPI remap function. When the remap offset is set, FlexSPI memory access changes the access address, adding the offset, as shown in this figure. For example, when the offset is set to 2 MB (200000h), the access to 8000_0000h via FlexSPI is remapped to 8020_0000h. Using this remap feature, the boot ROM can implement a dual image boot with two images. You must set the offset and remap size of the image in the CMPA region.

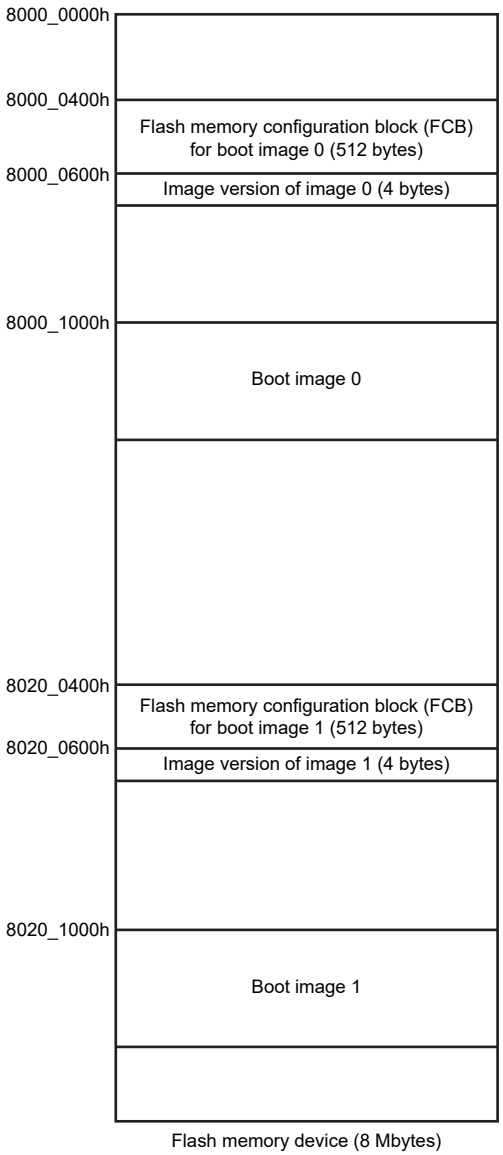


Figure 86. FlexSPI image remap (offset = 200000h)

15.3.1.2.5 Image remap offset and size configuration

The configurations related to FlexSPI dual image boot are allocated in CMPA word5 (0100_4014h) "FLEXSPI_BOOT_CFG [1]" word. See the following table and CMPA in the IFR table for details.

Table 237. Boot image 1 remap offset (CMPA: 0100_4014h)

Name that is filled	Bit offset	Width	Value	Description
FLEXSPI_IMAGE_OFFSET	7	10	*x*	*x* multiplied by 256 KB

Table 238. Boot Image size (FLEXSPI_BOOT_CFG [1])

Name that is filled	Bit offset	Width	Value	Description
FLEXSPI_REMAP_IMAGE_SIZE	17	4	0	Same size as that of the second image offset
			13	256 KB
			14	512 KB
			15	768 KB
			x	*x* multiplied by 1 MB

NOTE

FlexSPI remap size cannot exceed the start address of boot image 1.

15.3.1.2.6 Boot image version

Each boot image version is placed at offset 600h of the FlexSPI boot device (see Basic remapping function).

The image version uses 4 bytes—the lower 2 bytes are the real image version number, and the upper 2 bytes are the invert value of the lower 2 bytes of the image version. All other values that do not follow this rule are regarded as invalid values—if the boot image version is not valid, the boot ROM does not boot that image at all.

However, if the image version is all FFFF_FFFFh (the condition in which you do not program the image version), it is also considered as a valid value. When one of the image versions is FFFF_FFFFh and the other one is a valid image version, the boot ROM always first boots the image with the valid value, and not the one with image version as FFFF_FFFFh. If both image versions are FFFF_FFFFh, the boot ROM always boots the first image residing at the lower address in the FlexSPI NOR flash memory device.

A valid image version must always start from 1, which is FFFE_0001h.

The following table shows some examples of the boot image version.

Table 239. Boot image settings

Image version	Validity	Version number
FFFE_0001h	Valid	1
FFFD_0002h	Valid	2
0000_0003h	Invalid	Does not follow the image version rules
FFFF_FFFFh	Valid	Lower boot sequence
0000_FFFFh	Valid	Largest version, FFFFh

15.3.1.2.7 Ping-Pong boot flow

For FlexSPI boot, the boot ROM always tries to find the latest boot image to boot. If boot fails, the boot ROM tries to find an old boot image. If the boot still fails, it enters ISP boot mode. The following flowchart shows Ping-Pong boot flow.

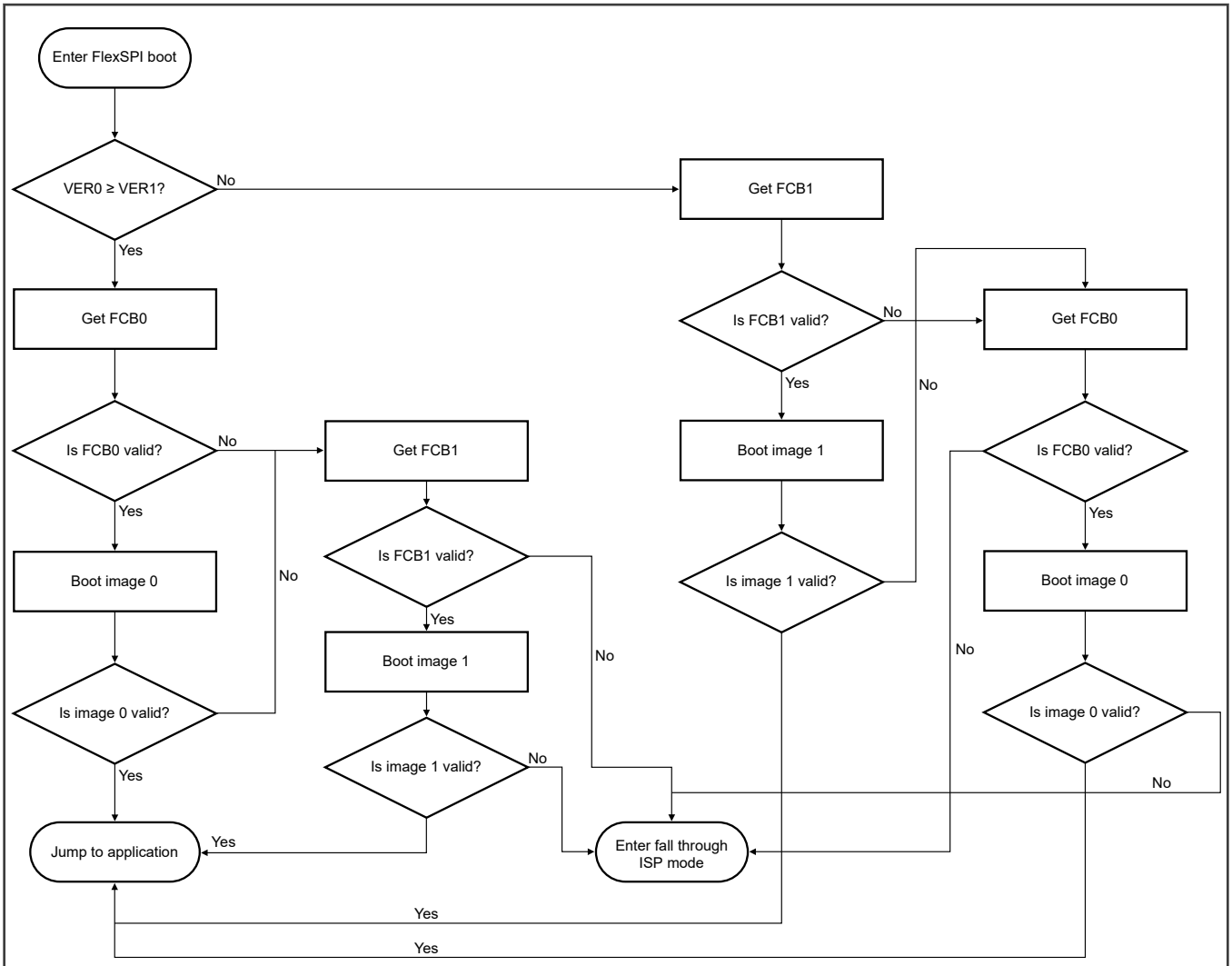


Figure 87. FlexSPI Ping-Pong boot flow

15.3.1.2.8 Configuring the FlexSPI NOR flash memory device

The boot ROM supports the booting of the image from an external FlexSPI NOR flash memory device. This section describes how to use the "blhost" tool to program the image into an external nonvolatile memory for booting. The "blhost" tool uses UART, SPI, I^2C, USB HID, and FlexCAN to communicate with the ROM code via ROM ISP mode.

See Table 227 for FlexSPI pin assignments of NOR flash memory connections.

15.3.1.2.8.1 Boot image programming

The boot ROM supports booting from different types of NOR flash memories such as Quad SPI, Octal SPI, and HyperFlash, which connect to FlexSPI pins that the boot ROM uses. For FlexSPI NOR boot, you must program FCB at offset 8000_0400h, and the boot image at offset 8000_1000h. The following sections provide information on how to configure and program the boot image into NOR flash memory for common NOR flash memory types.

15.3.1.2.8.2 Configuration options, settings, and parameters

15.3.1.2.8.2.1 Options

Table 240. FlexSPI serial NOR flash memory configuration options

Offset	Field	Description
0	Option0	See Table 241 for details.
4	Option1	Optional; effective only if the "option_size" field in option0 configuration is a nonzero value. See Table 242 for more information.

15.3.1.2.8.2.2 Option 0 field descriptions

Table 241. Option 0 field descriptions

Field	Bits	Description
tag	31:28	Tag of the configuration option, fixed to 0Ch
option_size	27:24	Size in bytes = (option size + 1) × 4 It is 0 only if option0 is required.
device_type	23:20	Device detection type: 0 – Read SFDP for SDR commands 1 – Read SFDP for DDR read commands 2 – Reserved 3 – HyperFlash 3V 4 – Macronix Octal DDR 6 – Micron Octal DDR 8 – Adesto EcoXiP DDR
query_pad	19:16	Data pads during Query command: (read SFDP or read MID) 0 – 1 2 – 4 3 – 8
cmd_pad	15:12	Data pads during flash memory access command: 0 – 1 2 – 4 3 – 8
quad_mode_setting	11:8	Quad SPI mode enable setting: 0 – Not configured 1 – Set bit 6 in Status register 1

Table continues on the next page...

Table 241. Option 0 field descriptions (continued)

Field	Bits	Description
		2 – Set bit 1 in Status register 2 3 – Set bit 7 in Status register 2 4 – Set bit 1 in Status register 2 by using the 0x31 command You use this setting to configure flash memory when it enters Quad SPI mode. You must reset the flash memory into stand SPI mode after the chip is reset because the boot ROM does not do that. **NOTE** This field is effective only if the chip is compliant with JESD216 (9 longword SFDP table).
misc_mode	7:4	Miscellaneous mode setting: 0 – Disabled 1 – Enable 0-4-4 mode for high random read performance 3 – Data Order Swapped mode (for MXIC OctaFlash memory only) 5 - Select the FlexSPI data sample source as internal loop back; see FlexSPI usage for more details 6 - Configure the FlexSPI NOR flash memory running at stand SPI mode **NOTE** Experimental feature only. Do not use in production: keep it as 0 for normal usage.
max_freq	3:0	Maximum flash memory operation speed: For SDR: 0 – 30 MHz (default) 1 – 50 MHz 2 – 60 MHz 3 – 75 MHz 4 – 100 MHz For DDR: 0 – 30 MHz 1 – 50 MHz

15.3.1.2.8.2.3 Option 1 field descriptions

Table 242. Option 1 field descriptions

Field	Bits	Description
Reserved	31:8	Reserved for future use

Table continues on the next page...

Table 242. Option 1 field descriptions (continued)

Field	Bits	Description
dummy_cycles	7:0	Dummy cycles for the Read command: 0 – Use detected dummy cycle Others – Use dummy cycles provided in the flash memory data sheet

15.3.1.2.8.2.4 Option settings

The following table shows typical FlexSPI serial NOR flash memory configuration option settings.

Table 243. FlexSPI serial NOR flash memory configuration option settings

FlexSPI mode name	Access type	Setting
Quad SPI NOR	Quad SPI SDR read	Option0 = C000_0002h (60 MHz)
Quad SPI NOR	Quad SPI DDR read	Option0 = C010_0002h (60 MHz)
HyperFlash 3V0	—	Option0 = C033_3002h (60 MHz)
MXIC OPI DDR (OPI DDR enabled by default)	—	C043_3001h (30 MHz)
Micron Octal DDR	—	Option0 = C060_0002h (60 MHz)
Micron OPI DDR	—	Option0 = C060_3002h (60 MHz)
Micron OPI DDR (DDR read enabled by default)	—	Option0 = C063_3002h (60 MHz)
Adesto OPI DDR	—	Option0 = C080_3002h (60 MHz)

15.3.1.2.8.2.5 Configuration parameters

```
                typedef struct _serial_nor_config_option
                {
          union
          {
            struct
            {
              uint32_t max_freq : 4;           // Maximum supported Frequency
              uint32_t misc_mode : 4;          // miscellaneous mode
              uint32_t quad_mode_setting : 4;  // Quad mode setting
              uint32_t cmd_pads : 4;           // Command pads
              uint32_t query_pads : 4;         // SFDP read pads
              uint32_t device_type : 4;        // Device type
              uint32_t option_size : 4;        // Option size, in terms of uint32_t, size =
(option_size + 1)
              uint32_t tag : 4;                     // Tag, must be 0x0C
            } B;
            uint32_t U;
          } option0;

          union
          {
            struct
```

```
            {
              uint32_t dummy_cycles : 8;     // Dummy cycles before read
              uint32_t status_override : 8;   // Override status register value during device mode
configuration
              uint32_t pinmux_group : 4;      // The pinmux group selection
              uint32_t dqs_pinmux_group : 4;  // The DQS Pinmux Group Selection
              uint32_t drive_strength : 4;    // The Drive Strength of FlexSPI Pads
              uint32_t flash_connection : 4;   // Flash connection option: 0 - Single Flash
connected to port A,  2 - Single Flash connected to Port B

            } B;
            uint32_t U;
          } option1;

               } serial_nor_config_option_t;
```

Code Listing 2. FlexSPI NOR flash memory device configuration parameters

15.3.1.2.8.3 Configuring, erasing, and programming via blhost

15.3.1.2.8.3.1 Configure via blhost

To configure a FlexSPI NOR flash memory device, the configuration parameter must be stored in RAM first. Consider Macronix MX25L25645G, which connects to the FlexSPI interface, as an example. The following figure shows the configuration parameter for FlexSPI, which is C000_0001h. You select a configuration parameter according to the flash memory type.

```
C:\blhost>blhost.exe -p com4 fill-memory 0x2000f000 4 0xc0000001
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Code Listing 3. Setting the FlexSPI configuration parameter in RAM

15.3.1.2.8.3.2 Read, erase, and program via blhost

Use the configuration parameter stored in RAM (see Configure via blhost) to configure FlexSPI, as shown in the following figure. After that, you can read, erase, and program the flash memory. For details related to blhost tool commands, use the help section of the blhost tool.

```
C:\blhost>blhost.exe -p com4 configure-memory 0x9 0x2000f000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Code Listing 4. Configuring FlexSPI using the parameter stored in RAM

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

```
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1) 100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Code Listing 5. Reading the NOR flash memory via the blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 flash-erase-region 0x08000400 0x200
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.
```

Code Listing 6. Erasing the NOR flash memory via the blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 write-memory 0x08001000 boot_image.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 5796 (0x16a4) bytes to the target.
Successful generic response to command 'write-memory'
(1/1) 100% Completed!
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 5796 of 5796 bytes.
```

Code Listing 7. Programming the boot image into NOR flash memory via the blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08001000 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
00 00 20 00 f9 d1 10 00 eb c8 10 00 5b cb 10 00
6f cc 10 00 bb cd 10 00 b7 d1 10 00 17 d3 10 00
a4 16 00 00 02 40 00 00 1d 26 cd c5 23 d3 10 00
9d d6 10 00 00 c0 10 00 9f d6 10 00 a1 d6 10 00
99 d4 10 00 a1 d4 10 00 a9 d4 10 00 b1 d4 10 00
b9 d4 10 00 c1 d4 10 00 c9 d4 10 00 d1 d4 10 00
d9 d4 10 00 el d4 10 00 e9 d4 10 00 f1 d4 10 00
f9 d4 10 00 01 d5 10 00 09 d5 10 00 11 d5 10 00
19 d5 10 00 21 d5 10 00 29 d5 10 00 31 d5 10 00
39 d5 10 00 41 d5 10 00 49 d5 10 00 51 d5 10 00
```

```
59 d5 10 00 61 d5 10 00 69 d5 10 00 71 d5 10 00
79 d5 10 00 81 d5 10 00 89 d5 10 00 91 d5 10 00
99 d5 10 00 a1 d5 10 00 a9 d5 10 00 b1 d5 10 00
b9 d5 10 00 c1 d5 10 00 c9 d5 10 00 d1 d5 10 00
d9 d5 10 00 e1 d5 10 00 e9 d5 10 00 f1 d5 10 00
f9 d5 10 00 01 d6 10 00 09 d6 10 00 11 d6 10 00
(1/1) 100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Code Listing 8. Reading the boot image back via the blhost tool

15.3.1.2.8.3.3 Generate and program FCB via blhost

Generate and program the FlexSPI NOR FCB into the flash memory for FlexSPI boot, with an FCB at offset 8000_0400h. You use this FCB to configure the FlexSPI interface when booting the image from an external NOR flash memory via the FlexSPI interface. You can generate the FCB using the previous FlexSPI configuration parameter (C000_0002h). Store the configured FCB and program parameter into RAM (see Code Listing 9). The boot ROM uses these in the next step to automatically generate and program the FCB into flash memory at offset 8000_0400h (see Code Listing 10).

```
C:\blhost>blhost.exe -p com4 fill-memory 0x2000f000 4 0xf000000f
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Code Listing 9. Storing the configured FCB parameter into RAM

```
C:\blhost>blhost.exe -p com4 configure-memory 0x9 0x2000f000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Code Listing 10. FCB generation and programming into the flash memory at offset 8000_0400h

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
46 43 46 42 00 04 01 56 00 00 00 00 01 03 03 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 01 04 02 00 00 00 00 00 00 00 00 00
00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ec 04 20 0a 00 1e 04 32 04 26 00 00 00 00 00 00
05 04 04 24 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
21 04 20 08 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(1/1) 100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Code Listing 11. Reading the FCB back via the blhost tool

15.3.1.2.8.4 Typical configuration parameters

You require different configuration parameters for different FlexSPI NOR flash memory types. The following table shows some typical flash memory configuration parameters.

Table 244. Typical NOR flash memory configuration parameters

Device vendor	Device type	Flash memory type	Configuration parameter	Flash memory mode	Comment
Adesto	Octal SPI flash memory	ATXP032	C080_3001h	DDR	• ATXP032 has three versions, REV A, REV B, and REV C. For REV A, the boot ROM only supports auto probe boot, the OTP fuse word. • You must configure FLEXSPI_BOOT_CFG[0] as 7h, with no need to configure FCB.
Cypress	HyperFlash memory	S26KS512S	C023_3001h	DDR	For HyperFlash boot, you must configure FLEXSPI_BOOT_CFG[0] as 20h before boot.
Macronix	Quad SPI flash memory	MX25L25645G	C000_0001h	SDR	—
Macronix	Quad SPI flash memory	MX25L25645G	C010_0001h	DDR	—
Macronix	Octal SPI flash memory	MX25UM51345G	C040_3001h	DDR	—
Micron	Octal SPI flash memory	MT35XL512ABA	C060_3001h	DDR	—

15.3.1.2.9 Programming the boot image into the flash memory

The following table lists the procedure and corresponding commands to program the boot image into the external flash memory with all data encrypted (see Configuring the FlexSPI NOR flash memory device for more information).

Table 245. FlexSPI NOR flash memory configuration

Step number	Step	Command to perform the step
1	Fill in the flash configuration parameter.	`blhost.exe -p com4 fill-memory 0x20020000 4 0xc0000001`
2	Enable the flash memory.	`blhost.exe -p com4 configure-memory 0x9 0x20020000`
3	Fill in the FCB generation parameters.	`blhost.exe -p com4 fill-memory 0x20020004 4 0xf000000f`
4	Program FCB at flash memory offset 8000_0400h.	`blhost.exe -p com4 configure-memory 0x9 0x20020004`
5	Erase the flash memory region from 8000_0000h to 8001_0000h.	`blhost.exe -p com4 flash-erase-region 0x80001000 0x10000`
6	Program the boot image into the flash memory from offset 8000_1000h.	`blhost.exe -p com4 write-memory 0x80001000 image.bin`

15.3.2 Secondary bootloader mode

Secondary boot mode can be enabled by setting CMPA[BOOT_SRC] as 2. The image loaded in the Bank1_IFR0 region (0x0100_8000 to 0x0100_FFFF) will be set as the primary boot mode, and the secondary boot image will be boot first after the device is reset. The secondary boot image can be plain, crc or signed image, but cannot be set as the SB file.

Also, the Bank1_IFR0 region (0x0100_8000 to 0x0100_FFFF) can be used as the Bank1_IFR0 recovery Boot area by setting the CMPA[REC_BOOT_SRC] as 1, then the Bank1_IFR0 region image will be set as the recovery boot image to boot when the primary image boot fail.

When the Bank1_IFR0 image is larger than the 32KB, user can also use the specific memory of the internal flash for the Bank1_IFR0 image usage based on the CMPA[REC_IMG_EXTn] corresponding bits, and each bit represent 32KB internal flash size. After the Bank1_IFR0 image boot, the specific memory of the internal flash and Bank1_IFR0 region will also be configured with the specific MBC as the bellow table. Suggest to use the last memory blocks of the internal flash for the Bank1_IFR0 image usage.

Based on the CMPA[OEM_BANK1_IFR0_PROT] setting, after the secondary boot image boot, the Bank1_IFR0 region will be configured with the settings as below.

Lifecycle	Boot mode			
	CMPA[OEM_BANK1_IFR0_PROT]	Secondary boot mode(SBL) & Internal flash(CMPA[REC_IMG_EXTn] corresponding block) MBC setting	Bank1_IFR0 recovery boot(OEM_REC) & Internal flash(CMPA[REC_IMG_EXTn] corresponding block) MBC setting	Not Secondary boot mode(SBL) and not Bank1_IFR0 recovery boot(OEM_REC)
Develop (0x3)	NA	GLBAC0	GLBAC0	GLBAC 0
Develop2 (0x7),	0	GLBAC4	GLBAC4	GLBAC 4
In-field (0xF),	1	GLBAC4		

Table continues on the next page...

Table continued from the previous page...

Lifecycle	Boot mode			
	CMPA[OEM_BANK1_IFR0_PROT]	Secondary boot mode(SBL) & Internal flash(CMPA[REC_IMG_EXTn] corresponding block) MBC setting	Bank1_IFR0 recovery boot(OEM_REC) & Internal flash(CMPA[REC_IMG_EXTn] corresponding block) MBC setting	Not Secondary boot mode(SBL) and not Bank1_IFR0 recovery boot(OEM_REC)
In-field Locked (0xCF), Field Return OEM (0x1F)	2	GLBAC2		
	3	GLBAC6		
	4	GLBAC4		
	5	Not allowed to set		GLBAC 2
	6			GLBAC 6
	7			GLBAC 7

Example 1: Under the LC = 7, when the Bank1_IFR0 region is used for the Secondary boot mode(CMPA[BOOT_SRC] as 2), the REC_IMG_EXT1[31] is set as 1 means the internal flash corresponding last block 32K (0x1F_8000 – 0x1F_FFFF) will be used for Secondary boot mode image usage. And set the CMPA[OEM_BANK1_IFR0_PROT] as 2, then the Bank1_IFR0 region and internal flash last block 32K (0x1F_8000 – 0x1F_FFFF) will be configured with GLBAC2 after the SBL boot.

Example 2: Under the LC = 7, when the Bank1_IFR0 region is used for the Bank1_IFR0 recovery boot mode(CMPA[REC_BOOT_SRC] as 1), and under the LC = 7 the REC_IMG_EXT1[31] is set as 1 means the internal flash corresponding last block 32K (0x1F_8000 – 0x1F_FFFF) will be used for Bank1_IFR0 recovery boot image usage, then the Bank1_IFR0 region and internal flash last block 32K (0x1F_8000 – 0x1F_FFFF) will be configured with GLBAC4 after the Bank1_IFR0 recovery boot.

15.3.3 ISP boot mode

The boot ROM supports an ISP which is mainly used for:

- Downloading the image (initial or updated) into the internal or external flash memory from the host.
- Provisioning the chip during production (configuring Secure mode or programming key data, ISP fall-through mode, and lock settings).

There are a number of conditions that can cause the boot ROM to enter ISP mode. By default, all entry methods are allowed, but CMPA fields can be used to restrict which ISP entry methods are allowed.

Table 246. ISP entry methods

ISP entry method	Description	CMPA field to allow/disallow
Boot fail	If the boot ROM does not locate a valid image, then the ROM can fall-through to ISP mode.	ISP_FT_ENTRY
API	The runBootloader API can be called from an application to request ISP mode entry.	ISP_API_ENTRY

Table continues on the next page...

Table 246. ISP entry methods (continued)

ISP entry method	Description	CMPA field to allow/disallow
Debug mailbox	The debug mailbox includes an Enter ISP mode command which can be used to request ISP entry from the debug port.	ISP_DM_ENTRY and ISP_CMD_EN[1]
ISP pin	If the ISP pin is asserted at reset, then the boot ROM can go straight to ISP mode without attempting to find a boot image.	ISP_PIN_ENTRY

1. ISP_CMD_EN is a field within the SOCU configuration. The combination of the fused DCFG_CC_SOCU plus the CFPA's DCFG_CC_SOCU and CMPA's CC_SOCU determine the access rights for debug commands including the Debug Mailbox Enter ISP mode command. The Debug Mailbox chapter is present in MCX N Security Reference Manual.

NOTE

The ISP_DISABLE fuse field can also be used to restrict ISP entry methods. For an ISP entry to work, it must be enabled by both the CMPA and fuse settings.

In ISP mode, the boot ROM can communicate over a variety of serial interfaces. By default, the device UART, I2C, SPI, CAN, and USB HID are supported. The CMPA's ISP_BOOT_IF can be used to select a specific serial interface to be used, or it can be left in the default Auto ISP mode which will scan all the available interfaces looking for activity. There are also fuses that can be used to disable ISP mode on specific interfaces:

- ISP_CAN_Dis
- ISP_USB1_Dis
- ISP_USB0_Dis
- ISP_I2C_Dis
- ISP_SPI_Dis
- ISP_UART_Dis

For ISP mode to work, the interface selected by ISP_BOOT_IF must be allowed by the ISP_xxx_Dis fuses. See the In-System Programming (ISP) chapter for more details on ISP protocol, packet types, commands, and operation for each serial communication type.

15.3.4 Recovery boot mode

The boot ROM supports a recovery boot from an external 1-bit SPI flash memory device or an image loaded into IFR0 (0x0100_8000 to 0x0100_FFFF). The table below summarizes the recovery boot options.

Table 247. Recovery boot options

Recovery boot type	Selected by	Image format
External 1-bit SPI flash	• CMPA REC_BOOT_SRC = 0b10 • Floating ISP pin	• SB3.1 image • Load-to-RAM image
Bank1_IFR0 recovery boot(OEM_REC) (0x0100_8000 to 0x0100_FFFF)	CMPA REC_BOOT_SRC = 0b01	XIP image

See SPI flash memory recovery for more information.

NOTE

The default pins used for 1-bit SPI flash recovery boot overlap with the FlexSPI pins. This was done intentionally to allow a memory to be used as a recovery device and also to be used for FlexSPI operation later.

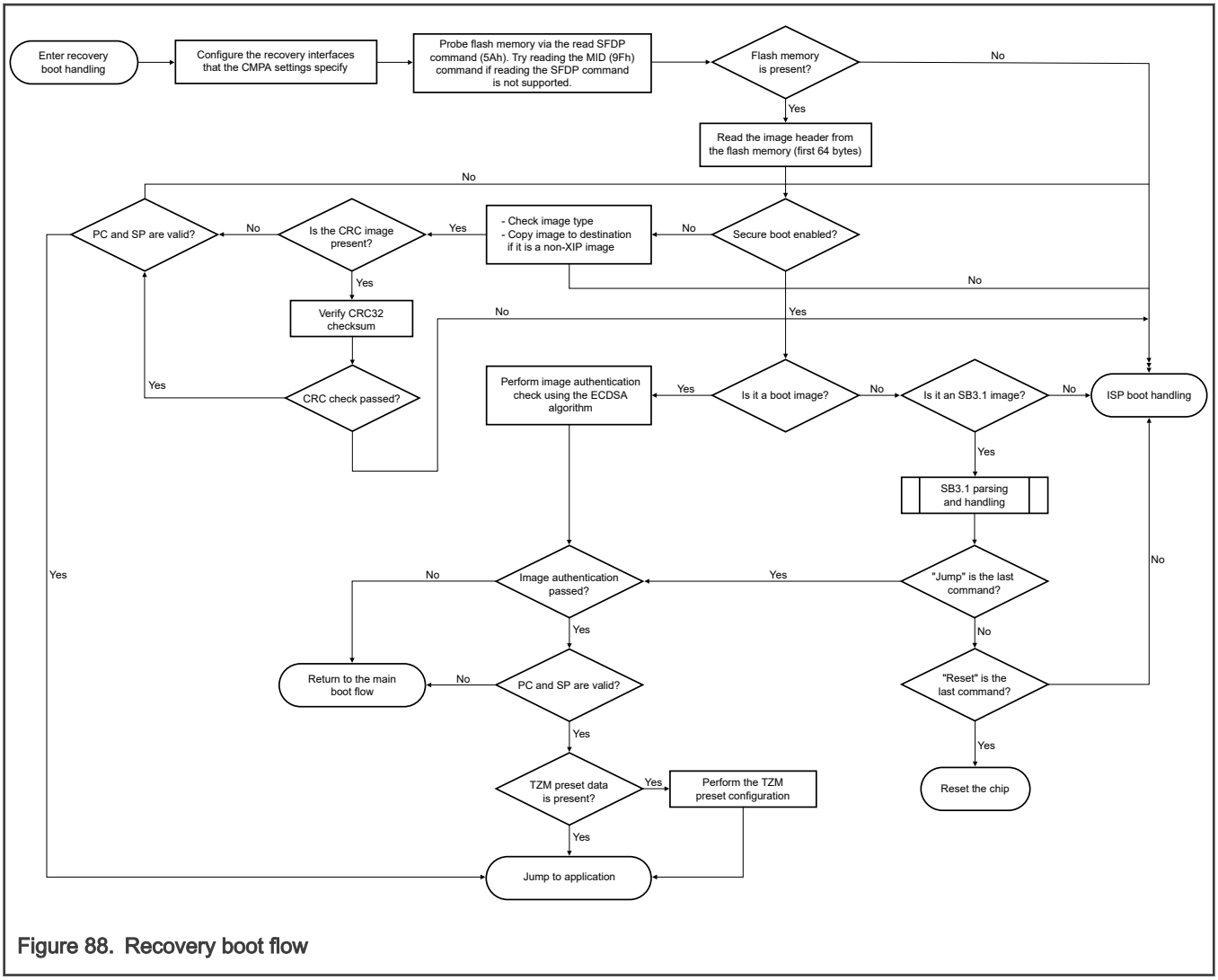


Figure 88. Recovery boot flow

15.4 External memory support

The following table describes the external memory devices, with their memory identifiers, that the boot ROM ISP command supports.

To use an external memory device correctly, the boot ROM enables specific external memory devices, with their corresponding configuration profiles, using a preassigned memory identifier. Without enabling these external memory devices, you cannot access them using the boot ROM ISP command.

Table 248. Memory IDs for external memory devices

Memory identifier	External memory device
09h	Serial NOR over the FlexSPI module
110h	Serial NOR and EEPROM over the SPI module

15.4.1 Serial NOR flash memory through FlexSPI

15.4.1.1 FlexSPI NOR FCB

The boot ROM supports the read, write, and erase functions for external serial NOR flash memory devices via FlexSPI. Before accessing these devices, you must configure the FlexSPI module properly, using a simplified FlexSPI NOR configuration block or a complete 512-byte FlexSPI NOR configuration block. See Table 249 for settings.

The boot ROM can generate the 512-byte FlexSPI NOR configuration block based on the simplified flash memory configuration option block for most serial NOR flash memory devices in the market.

To customize the lookup table (LUT) sequence for a specific chip, you must enable the “lutCustomSeqEnable” field and fill in the corresponding “lutCustomSeq” field. See Table 250 for the predefined LUT index for serial (SPI) NOR.

Table 249. FlexSPI NOR FCB

Field	Offset (hex)	Size (bytes)	Description
tag	000	4	Configuration block tag Must be set to 4246_4346h
version	004	4	Configuration block that the boot ROM uses internally; its value is fixed to 5601_0400h
Reserved	008	4	Reserved for future use
readSampleClkSrc	00C	1	Read sampling clock source option: 0 – Internal sampling 1 – Loopback from DQS pad 2 – Loopback from SCK pad 3 – External DQS signal
csHoldTime	00D	1	CS hold time in terms of flash memory clock cycles Recommended value is 3.
csSetupTIme	00E	1	CS set up time in terms of flash memory clock cycles Recommended value is 3.
columAddressWidth	00F	1	Column address width: • 3 for HyperFlash • 0 for other flash memory devices
deviceModeCfgEnable	010	1	Enables the Device mode configuration sequence
deviceModeType	011	1	Argument for Device mode configuration For example, Quad enable setting in Status register 2 for some Quad SPI flash memory devices
waitTimeCfgCommands	012	2	Wait time in terms of 100 µs for the Device mode configuration command

Table continues on the next page...

Table 249. FlexSPI NOR FCB (continued)

Field	Offset (hex)	Size (bytes)	Description
deviceModeSeq	014	4	Device mode configuration sequence: Byte 0 – Number of required sequences Byte 1 – Sequence index
deviceModeArg	018	4	Argument for Device mode configuration For example, Quad enable setting in Status register 2 for some Quad SPI flash memory devices
configCmdEnable	01C	1	Configuration command enable: 0 – Ignores configuration command 1 – Enables configuration Command
configModeType	01D	3	Configuration command types; supports up to three types Each type is combined with a configuration command sequence.
configCmdSeqs	020	12	Configuration command types; supports up to three sequences
Reserved	02C	4	Reserved for future use
configCmdArgs	030	12	Configure command argument; supports up to three arguments
Reserved	03C	4	Reserved for future use
controllerMiscOption	040	4	Miscellaneous controller configuration options: Bit 0 – Differential clock enable; set to 1 for the HyperFlash 1V8 device, and set to 0 for other devices Bit 3 – WordAddressableEnable; set to 1 for HyperFlash and set to 0 for other devices Bit 4 – SafeConfigFreqEnable; set to 1 for configuring the device with a safe frequency Bit 6 – DDR mode enable; set to 1 if DDR read is expected Other bits – Reserved; set to 0
deviceType	044	1	Device type: 1 - Serial NOR
sflashPadType	045	1	Data pad used in Read command: 1 – Single pad 2 – Dual pads

Table continues on the next page...

Table 249. FlexSPI NOR FCB (continued)

Field	Offset (hex)	Size (bytes)	Description
			4 – Quad pads 8 – Octal pads
serialClkFreq	046	1	SDR: 0 – 30 MHz 1 – 50 MHz 2 – 60 MHz 3 – 75 MHz 4 – 100 MHz DDR: 0 – 30MHz 1 – 50MHz
lutCustomSeqEnable	047	1	LUT customization enable that is required if you cannot perform the program or erase operations using one LUT sequence (currently only applicable to HyperFlash)
Reserved	048	8	Reserved for future use
sflashA1Size	050	4	Size of flash memory connected to A1
sflashA2Size	054	4	Size of flash memory connected to A2
sflashB1Size	058	4	Size of flash memory connected to B1
sflashB2Size	05C	4	Size of flash memory connected to B2
csPadSettingOverride	060	4	Pad override value for CS pin Use this value (if nonzero) to configure the CS pin; otherwise use the default ROM setting
sclkPadSettingOverride	064	4	Serial clock pad setting override value
dataPadSettingOverride	068	4	Data pad setting override value
dqsPadSettingOverride	06C	4	DQS pad setting override value
timeoutInMs	070	4	Timeout value in terms of ms to terminate the busy check
commandInterval	074	4	CS deselect interval between two commands
dataValidTime	078	4	Clock edge to data valid time
busyOffset	07C	2	Busy offset; valid value: 0–31

Table continues on the next page...

Table 249. FlexSPI NOR FCB (continued)

Field	Offset (hex)	Size (bytes)	Description
busyBitPolarity	07E	2	Busy flag polarity: 0 – Indicates that the busy flag is set to 1 when the flash memory device is busy 1 – Indicates that the busy flag is set to 0 when the flash memory device is busy
lookupTable	080	256	16 LUT sequences Each sequence consists of four words (see the "FlexSPI" chapter for more information).
lutCustomSeq	180	48	Customizable LUT sequences
Reserved	1b0	16	Reserved for future use
pageSize	1C0	4	Page size of flash memory
sectorSize	1C4	4	Sector size of flash memory
ipcmdSerialClkFreq	1C8	1	Serial clock frequency for IP command: 0 – The same with Read command Others – The same definition as serialClkFreq
isUniformBlockSize	1C9	1	Indicates whether the sector size is the same as block size: 0 – No 1 – Yes
isDataOrderSwapped	1CA	1	Indicates whether the data order (D0, D1, D2, D3) is swapped (D1, D0, D3, D2)
Reserved	1CB	1	Reserved for future use
serialNorType	1CC	1	Serial NOR flash memory type: 0, 1, 2, or 3
needExitNoCmdMode	1CD	1	Exit NoCmd mode before other IP commands as needed
halfClkForNonReadCmd	1CE	1	Half the serial clock for nonread command: true or false
needRestoreNoCmdMode	1CF	1	Restore NoCmd mode after IP command execution as needed
blockSize	1D0	4	Flash memory block size
flashStateCtx	1D4	4	Flash memory state context

Table continues on the next page...

Table 249. FlexSPI NOR FCB (continued)

Field	Offset (hex)	Size (bytes)	Description
Reserved	1D8	40	Reserved for future use

15.4.1.2 LUT index pre-assignment for FlexSPI NOR

Table 250. LUT index pre-assignment for FlexSPI NOR

Name	Index in LUT	Description
Read	0	Read command sequence
ReadStatus	1	Read status command
ReadStatusXpi	2	Read status command under OPI mode
WriteEnable	3	Write enable command sequence
WriteEnableXpi	4	Write enable command under OPI mode
EraseSector	5	Erase sector command
EraseBlock	8	Erase block command
PageProgram	9	Page program command
ChipErase	11	Full chip erase
ExitNoCmd	15	Exit no command mode as needed
Reserved	6, 7, 10, 12, 13, 14	All reserved indexes can be freely used for other purposes

15.4.2 FlexSPI NOR configuration option block settings

The FlexSPI NOR configuration option block is organized by a 4-bit unit and is expandable (see Table 251 and Table 252 for the current definitions of the block).

The boot ROM detects FCB via reading an SFDP command, which most JESD216(A/B)-compliant flash memory devices support. However, JESD216A/B only defines the dummy cycles for Quad SPI SDR read. To get the dummy cycles for DDR or DTR read mode, the boot ROM supports auto-probing by writing test patterns to offset 200h on the external flash memory devices. For optimal timing, readSampleClkSrc is set to 1 for flash memory devices that do not support externally provided DQS pad inputs. It is set to 3 for flash memory devices that support externally provided DQS pad inputs such as Octal SPI flash memory or HyperFlash NOR flash memory. The FlexSPI_DQS pad is not used for other purposes.

Table 251. FlexSPI NOR configuration option 0 block

Field: Option0 | Offset: 0 Details							
Tag[31:28]	Option size[27:24]	Device detection type[23:20]	Query CMD pad(s) [19:16]	CMD pads(s) [15:12]	Quad SPI enable type[11:8]	Misc mode[7: 4]	Max freq[3:0]
Ch (fixed)	Size in bytes = (option size + 1) × 4 (provides scalability for future use)	Software-defined device types used for configuration block auto-detection 0: Quad SPI SDR 1: Quad SPI DDR 2: HyperFlash 1V8 3: HyperFlash 3V 4: MXIC OPI DDR 6: Micron OPI DDR 7: Micron Octal SDR 8: Adesto OPI DDR 9: Adesto EcoXiP SDR	Command pads (1, 4, and 8) for the SFDP command 0: 1 2: 4 3: 8	Command pads (1, 4, and 8) for the flash memory device 0: 1 2: 4 3: 8	Quad enable sequence 1: QE bit is bit 6 in StatusReg1 2: QE bit is bit 1 in StatusReg2 3: QE bit is in bit 7 in StatusReg2 4: QE bit is bit 1 in StatusReg2; enable command is 0x31	Miscellaneous mode for the selected flash memory types **NOTE** Experimental feature only. Do not use in production: keep it as 0 for normal usage. 0: Not enabled 1: Enable 0-4-4 mode for high random read performance 3: Data orders wrapped mode (for MXIC OctaFlash only) 5: Select the FlexSPI data sample source as internal loop back; see FlexSPI usage for more 6: Configure the FlexSPI NOR flash memory running at stand SPI mode	Maximum flash memory operation speed 0: Don't change the FlexSPI clock setting

Table 252. FlexSPI NOR configuration option 1 block (optional)

Field: Option1 (optional) | Offset: 4 Details			
flash_connection[31:28]	Reserved[27:16]	status_override[15:8]	Dummy cycle[7:0]
Flash connection option (FlexSPI port A or B) 0: Single flash memory connected to port A 2: Single flash memory connected to port B		Override status register value used for Device mode configuration	Dummy cycles for SDR and DDR read commands 0: Use detected dummy cycles Other values: Use dummy cycles provided in the flash memory data sheet

NOTE

CMD pad(s) value is:

- Always 0h for devices that work under 1-1-4, 1-4-4, 1-1-8, or 1-8-8 mode.
- 2 for devices that only support 4-4-4 mode for high performance.
- 3 for devices that only support 8-8-8 mode for high performance.

15.4.3 Typical use cases for the FlexSPI NOR configuration block

Table 253. Use cases for the FlexSPI NOR configuration block

FlexSPI mode name	Access type	Setting
Quad SPI NOR	Quad SDR read	Option0 = C000_0002h (60 MHz)
Quad SPI NOR	Quad DDR read	Option0 = C010_0002h (60 MHz)
HyperFlash 3V0	—	Option0 = C033_3002h (60 MHz)
MXIC OPI DDR	OPI DDR enabled by default	Option0 = C043_3002h (60 MHz)
Micron Octal DDR	DDR read	Option0 = C060_0002h (60 MHz)
Micron OPI DDR	DDR read	Option0 = C060_3002h (60 MHz)
Micron OPI DDR	DDR read enabled by default	Option0 = C063_3002h (60 MHz)
Adesto OPI DDR	DDR read	Option0 = C080_3002h (60 MHz)

15.4.4 Programming a serial NOR flash memory device using the FlexSPI NOR configuration option

The boot ROM supports generating the complete FCB using the configure-memory command. It also supports programming the generated FCB at the start of the flash memory by using a specific option, "0xF000000F." The following table shows an example of how to configure and access HyperFlash (assuming it is a blank HyperFlash device). See Configuring the FlexSPI NOR flash memory device for more information.

Table 254. Programming a serial NOR flash memory device using the FlexSPI NOR configuration option

Step number	Step	Command to perform the step
1	Write the option block to SRAM address 2002_000h.	`blhost -u <PID,VID> - fill-memory 0x20020000 0x04 0xc0333007`
2	Configure HyperFlash using the option block.	`blhost -u <PID,VID> - configure-memory 0x09 0x20020000`
3	Write specific option to SRAM address 2002_000h.	`blhost -u <PID,VID> - fill-memory 0x20020000 0x04 0xf000000f`
4	Program FCB to the offset 400h of the HyperFlash device.	`blhost -u <PID,VID> - configure-memory 0x09 0x20020000`
5	Erase the HyperFlash device from address of specified size.	`blhost -u <PID,VID> – flash-erase-region <addr> <size>`
6	Write the image to the address.	`blhost -u <PID,VID> - write-memory <addr> image.bin`

15.4.5 1-bit serial NOR flash memory through SPI

15.4.5.1 Manual and Auto configuration methods

The boot ROM supports programming 1-bit SPI NOR flash memory devices that support a 3-byte address read, 03h, or 4-byte address read, 13h, via the flash memory configuration option block. It defines the SPI clock frequency as 48 MHz.

The boot ROM supports either a manually configured option or an auto-detected option. When using the manually configured option, you must specify all the flash memory information (flash size, sector size, page size, and so on) in the option block. When using the auto-detected option, which is only supported on devices that are JESD216 compliant, the boot ROM is able to detect the flash memory information via the read SFDP (5Ah) command. Using this command, you can set all the flash memory information to zeros. Table 255 shows details required for configuring the SPI NOR flash memory by using these methods. The read page (03h) is used for the default read command. If the flash memory size is greater than 16 MB (detected by the read SFDP command), the 13h command is used for the page read. If the SFDP command is not supported, the boot ROM uses the read MID (9Fh) to detect whether a device is connected to it.

15.4.5.2 SPI NOR configuration option block settings

Table 255. SPI NOR configuration option block settings (Option0 field settings)

Tag[31:28]	Reserved[27:16]	Flash memory info set[15:12]	Flash memory size[11:8]	Sector size[7:4]	Page size[3:0]
Fixed at Ch		Memory type used to configure and access the NOR flash memory: 0 – Manual (selects flash memory parameter via the flash memory data sheet) 2 – Auto (detects flash memory parameter via the SFDP table)	Memory capacity of the NOR flash memory device: 1 – 1 MB 2 – 2 MB 3 – 4 MB 4 – 8 MB 5 – 16 MB 6 – 32 MB 7 – 64 MB 8 – 64 MB 9 – 128 MB 10 – 256 MB	Sector size of the NOR flash memory device: 0 – 4 KB 1 – 8 KB 2 – 32 KB 3 – 64 KB 4 – 128 KB 5 – 256 KB	Page size of the NOR flash memory device: 0 – 256 bytes 1 – 512 bytes 2 – 1 KB 3 – 128 KB

See Table 256 for information about how to program a 1-bit serial NOR flash memory device via the SPI NOR configuration option block.

15.4.5.3 Programming via the SPI NOR configuration option block

The boot ROM supports programming the serial NOR flash device via the SPI interface using the SPI NOR configuration option block. The following table shows an example of how to configure or program the option block. Before writing the configuration option block, you select the SPI peripheral index by writing the value C070_2000h. (SPI index is the SPI interface used to access serial NOR flash memory.)

Table 256. Programming a 1-bit serial NOR flash memory device via the SPI NOR configuration option block

Step number	Step	Command to perform the step
1	Write the option block to SRAM address 2002_0000h; the SPI instance is 7.	`blhost -p comxx - fill-memory 0x20020000 0x04 0xc0702000`
2	Configure serial NOR flash memory using the option block.	`blhost -p comxx - configure-memory 0x110 0x20020000`
3	Erase the serial NOR flash memory device from address of specified size.	`blhost -p comxx - flash-erase-region <addr> <size> 0x110`
4	Write the recovery boot image or SB3 file to the address in the external 1-bit flash memory.	`blhost -p comxx - write-memory <addr> image.bin 0x110`

15.5 eFuse definitions

The boot ROM supports the fuse block and defines the logical fuse layout provided in the following table, with exported APIs for fuse read and program.

Chapter 16
Debug

16.1 Introduction

This chapter describes the debug architecture of this chip. See Figure 89.

16.1.1 Features

- Supports Joint Test Access Group (JTAG) and Serial Wire Debug (SWD); the primary CM33 CPU supports ETB with 2 KB RAM
- Is based on the Arm CoreSight™ architecture that is adapted for software tracing and debugging
- Implements SWD and Debug Access Port (DAP) to support debugger tools
- Supports the following for the subsystem that uses the Cortex-M33 core:
  - Instrumentation Trace Macrocell (ITM)
  - Data Watchpoint and Trace (DWT)
  - Breakpoint Unit (BPU)
  - Software trace and full instruction trace
- Provides a 4-bit trace port interface unit (TPIU) and 1-bit SWO trace port to efficiently access the Cortex-M33 core's trace information from the system
- Supports cross-triggered interface (CTI). One Cortex-M33 core can halt or restart the other one. The primary Cortex-M33 core's CTI channel input has connections with ETB debug halt, Baseband Signal Processor (BSP) with 32-bit data path and 32-bit data memories (BSP32) restart, and BSP32 break signals.

16.1.2 Architecture

The following figure shows the system level debug architecture.

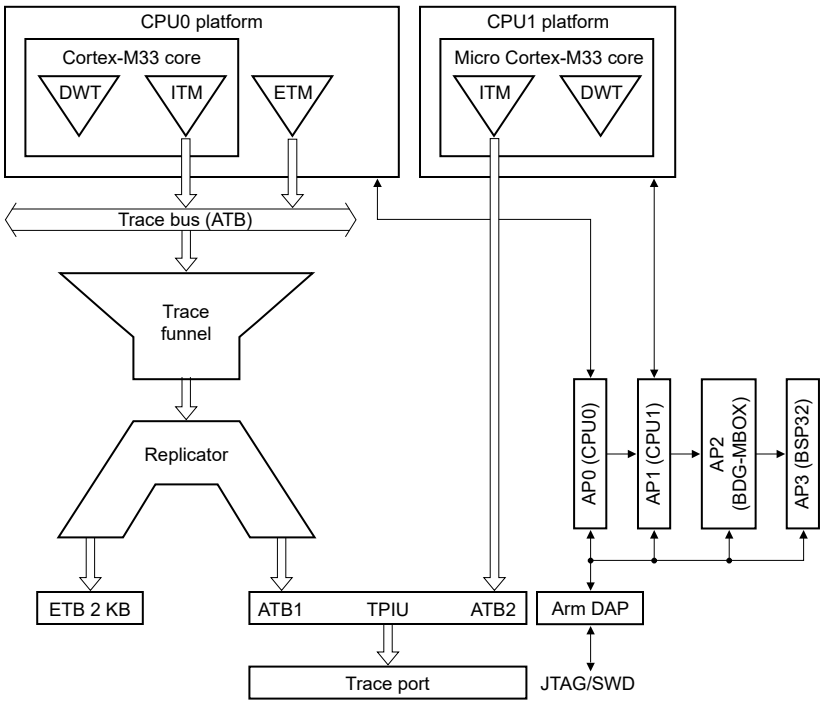


Figure 89. Debug architecture

16.2 Test and debug port connectivity

The following figure shows the connectivity of TAP and DAP on this device.

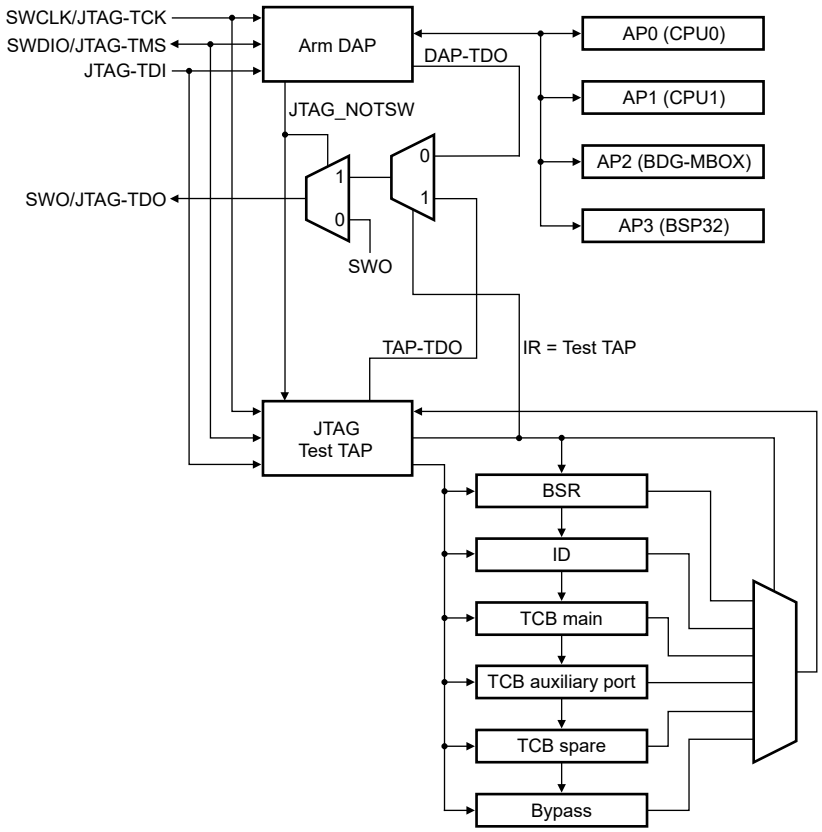


Figure 90. TAP and DAP connectivity

The following table summarizes the JTAG and SWD signals.

Table 257. JTAG and SWD signals description

Pin name	JTAG		SWD		Internal pullup or pulldown
	Type	Description	Type	Description	
TMS or SWDIO	I/O	Mode selection	I/O	Data	Pullup
TCLK or SWCLK	I	Clock	I	Clock	Pulldown
TDI	I	Data input	--	--	Pullup
TDO	O	Data output	--	--	N/A

16.2.1 JTAG

16.2.1.1 Instruction registers

JTAG TAP connects to the Arm DAP controller in parallel, that is, the JTAG-DP port of the DAP. The length of the instruction register (IR) is 4-bits. The JTAG TAP IR codes overlay the DAP controller IR codes. JTAG TAP uses 12 instructions, and the DAP uses the remaining four instructions. The TAP (TDO) outputs are multiplexed based on the selected IR code. This design is fully JTAG compliant and appears to the JTAG chain as a single TAP.

The following table provides a list of IR codes for the system JTAG controller.

Table 258. JTAG TAP

Code	JTAG IR
0000b	IR_EXTEST
0001b	SAMPLE/PRELOAD
0010b	IR_HIGHZ
0011b	IR_CLAMP
0100b	IR_IDCODE
0101b	Reserved
0110b	Reserved
0111b	Reserved
1000b	ARM-DAP IR_ABORT
1001b	Reserved
1010b	ARM-DAP IR_DPACC
1011b	ARM-DAP IR_APACC
1100b	Reserved
1101b	Reserved
1110b	ARM-DAP IR_DAP_ID
1111b	ARM-DAP BYPASS

16.2.1.2 Switching JTAG mode to SWD mode

JTAG mode is the default connection of the JTAG/SWD port. The following sequence of events switch the debug port to SWD mode:

1. Send more than 50 TCLK cycles with TMS/SWDIO = 1.
2. Send the 16-bit sequence on TMS/SWDIO = 0111_1001_1110_0111 (MSB transmitted first).
3. Send more than 50 TCLK cycles with TMS/SWDIO = 1.

16.2.2 DAP

DAP is a standard Arm CoreSight component that provides multiple master driving ports, all accessible and controlled through a single external interface port to provide system-wide debug. The following table lists the DAP IR codes:

Table 259. DAP IR Codes

Code	DAP IR
1000b	ABORT
1010b	DPACC
1011b	APACC
1110b	IDCODE

The DAP offers AHB and APB master interfaces to access system buses. It also exports the internal DAP bus to extend the access ports as per the system requirement. It offers JTAG AP to add auxiliary JTAG TAPs. For more information on DAP or JTAG chips under JTAG AP, see *Arm Debug Interface v5 Architecture Specification* on arm.com

The AHB AP's AHB transfers are burst size of 1 only with no out-of-order transactions and no multiple outstanding accesses. Use the APB AP to access debug components of the system trace such as TPIU and ETB.

Use the exported DAP bus to host AHB AP (integrated as part of the Cortex-M33 core integrations). Select different access ports in the DAP based on the APSEL value set in the SELECT register of SWJ-DP.

The following figure shows the DAP configuration on this chip.

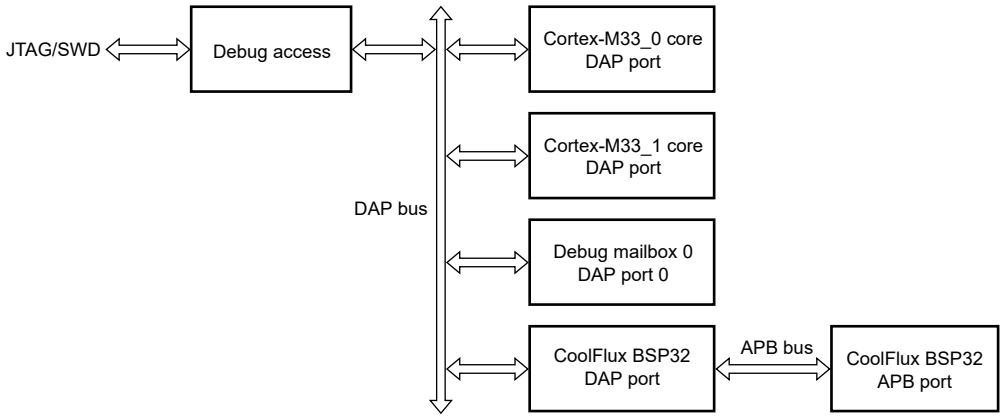


Figure 91. DAP configuration

16.3 Debug ROM table

The debug ROM table holds information about different debug components and help identify them. In this chip, this table resides in the CM33 core and contains entries for CM33 debug components.

Table 260. Debug ROM table

Processor		Start address
CPU0 ROM		E00FE000h
	Cortex-M33 core processor ROM	E00FF000h
	TPIU	E0040000h
	Trace funnel	E0044000h
	ETB	E0045000h
CPU1 ROM		E00FF000h

NOTE

Both CPU0 and CPU1 share the same TPIU inside CPU0 at E0040000h, which only CPU0 can program via its Private Peripheral Bus (PPB).

16.4 Cortex-M33 core debug

This section discusses the debug mechanisms associated with the Cortex-M33 core processor.

16.4.1 Software debug and trace

See Figure 89 for more details on the data flow of software debug and trace information for the Cortex-M33 core on this chip.

16.4.2 DWT

The Cortex-M33 core's DWT module generates trace information in the core. You can program the module's four independent watchpoints for comparing data, address, or program counter (PC) to trigger ETM or to function as a hardware watchpoint. You can configure DWT to generate PC samples at defined intervals and generate interrupt event information. DWT also provides periodic requests for protocol synchronization to the ITM and TPIU.

16.4.3 BPU

The Cortex-M33 core's BPU module consists of up to eight instruction address comparators. It provides breakpoint functionality on all instructions fetched across the entire address range in which you can locate the code.

16.4.4 ITM

ITM generates trace information as packets, and there are multiple sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The sources in decreasing order of priority are as follows:

- Software trace: Your software can write console messages directly to ITM stimulus ports and output them to the host as trace packets.
- Hardware trace: DWT generates these packets, and the ITM outputs them.
- Timestamping: ITM can generate timestamp packets that are inserted into the trace stream to help the host debugger to figure out the timing of events. Timestamp generation occurs relative to packets. ITM contains a 21-bit counter to generate the timestamp.

Trace data from ITM is forwarded to either the TPIO or SWO and streamed out via the trace port.

16.4.5 ETM

Instruction trace, also known as ETM trace, is a continuously collected sequence of every executed instruction for a selected portion of the application. ETM generates trace packets and sends them to the trace bus. It does not output every address or instruction that the processor has executed; it generates compressed information about the program flow and outputs full addresses as per requirement (for example, if a branch takes place). Because the debugger knows the application code image, the debugger can then reconstruct the full instruction sequence from the trace data.

16.5 In-system programming access port (ISP-AP)

ISP-AP provides a useful interface between the debugger and the core, and thus it controls the debug information.

16.6 CTI

The following table defines CTI input triggers.

Table 261. CTI input triggers

Signal	Description	Connection	Acknowledge, handshake
CTITRIGIN[7]	--	ETM to CTI	Pulsed
CTITRIGIN[6]	--		
CTITRIGIN[5]	ETM event output 1		

Table continues on the next page...

Table 261. CTI input triggers (continued)

Signal	Description	Connection	Acknowledge, handshake
CTITRIGIN[4]	ETM event output 0 or comparator output 3	ETM or processor to CTI	
CTITRIGIN[3]	DWT comparator output 2	Processor to CTI	
CTITRIGIN[2]	DWT comparator output 1		
CTITRIGIN[1]	DWT comparator output 0		
CTITRIGIN[0]	Processor halted		

The following table defines CTI output triggers. The Cortex-M33 cores support CTI. One Cortex-M33 core can halt or restart the other one. The CPU0 CTI channel input has connections with ETB debug halt, BSP32 restart, and BSP32 break signals:

- CPU0 CTICHIN = {CPU1 CTICHOUT[1], ETB debug halt, BSP32 restart, BSP32 break}
- CPU1 CTICHIN = {CPU0 CTICHOUT[1], 000}

Table 262. CTI output triggers

Signal	Description	Connection	Acknowledge, handshake
CTITRIGOUT[7]	ETM event input 3	CTI to ETM	Pulsed
CTITRIGOUT[6]	ETM event input 2		
CTITRIGOUT[5]	ETM event input 1 or MTB trace stop	CTI to ETM or MTB	
CTITRIGOUT[4]	ETM event input 0 or MTB trace stop		
CTITRIGOUT[3]	Interrupt request 1	CTI to system	Acknowledged by writing to the CTIINTACK register in ISR
CTITRIGOUT[2]	Interrupt request 0		
CTITRIGOUT[1]	Processor restart	CTI to processor	Processor restarted
CTITRIGOUT[0]	Processor debug request		Acknowledged by the debugger writing to the CTIINTACK register in ISR

16.7 Low-power debug

CMC.DBGCTL[SOD] controls low-power debug. Writing 1 disables the debugger (debug power-up acknowledge negates) when the Cortex-M33 core sleeps, and the chip fully enters Low-Power mode. When CMC.DBGCTL[SOD] = 0, the Cortex-M33 clocks remain enabled while the Cortex-M33 core sleeps. In this condition, the debug powerup request is asserted to maintain the debug connection. The chip does not enter Low-Power mode completely. Your software can modify the value of DBGCTL[SOD] before each low-power entry. Supported modes include:

- Clock-Gated mode
- Power-Gated mode

16.7.1 Clock-Gated mode

If the SWJDAP debug powerup request is high, and CMC.DBGCTL[SOD] = 0, when the system attempts to enter STOP mode, the DAP clock and the FCLK continue to run to support core register access and trace. This happens when the system attempts to enter STOP mode. In this case, the debug module has access to core registers but not to modules that are clock-gated.

16.7.2 Power-Gated mode

You must implement SWJDP in Wake domain and can retain the JTAG/SWD status when the chip enters Deep Sleep or Power-Down mode. In this way, the debugger remains attached to the SWD/JTAG, but the debug powerup acknowledge remains negated to indicate that the debugger cannot connect to any of the debug registers in the Cortex-M33 core. When the chip enters Deep Power-Down mode, JTAG/SWD is reset. Debug access is disabled on every warm reset and remains disabled until the completion of boot ROM execution. This includes a wake-up from Power-Down or Deep Power-Down mode. When wake-up from Power-Down or Deep Power-Down mode occurs, the boot ROM supports either normal boot flow or low-power wake-up flow. In normal boot flow, boot ROM can configure debug based on either debug authentication results or fuse life cycle bit configuration. In low-power wake-up flow, boot ROM configures debug only based on life cycle bit configuration. See the ROM chapters on how to configure debug.

16.8 Debug authentication

The Armv8-M processor implements debug authentications using the CoreSight signals, such as DBGEN, NIDEN, SPIDEN, and SPNIDEN. This chip implements debug authentication to control the state of SPIDEN and SPNIDEN signals from Arm using the ROM and debug mailbox (DM). A 2-bit field in the protected flash region (called either IFR or PFR) specifies the secure debug enable control, and the field values perform the following functions:

- 00: Enables secure debug always (factory default state)
- 01: Enables secure debug using "image authentication", but does not write lock the register before entering the user code
- 10: Enables secure debug using "image authentication" and write locks the register before entering the user code (a blank or corrupted flash memory disables secure debug until the next POR)
- 11: Disables secure debug and write locks the register

Chapter 17
System Controller (SYSCON)

17.1 Chip-specific SYSCON information

Table 263. Reference links to related information

Topic	Related module	Reference
Full description	SYSCON	SYSCON
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

The reset values of some registers may change depending on the device's boot settings.

17.1.1 Module instances

This device has one instance of SYSCON module, SYSCON0.

NOTE

The SYSCON module includes some security functionality that is not described here. Refer to the MCX N Security RM for more details.

17.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

17.1.3 Configuration

Configure the SYSCON block as follows:

- No clock configuration is needed. The clock to the SYSCON block is always enabled.
- The SYSCON block controls use of the CLKOUT pin, which must also be configured through Port control module.

17.2 Overview

The SYSCON module provides controls and configurations on the system and peripherals for the multiple functions.

17.2.1 Features

The SYSCON module supports the following features and configurations:

- System and bus configuration
  - AHB matrix priority
  - CPUs control and status
  - CPU and DSP debug access
  - CPU LPCAC control

  - NMI source select
  - Calibrate system tick timer
  - Boot control
- System/Peripherals - clock select and control
  - Allows enabling and selection of clocks to individual peripherals and memories
  - Allows configuration of clock dividers
- Memory configuration
  - ROM access
  - RAM ECC
  - FLASH cache, bus error
- Gray-2-Binary converter
  - Allows decoding gray value coming from OS Event Timer
- Peripherals configuration
  - SmartDMA interrupt hijack
  - PWM0/1 control
  - Ethernet PHY selection and flow control
  - CTIMER global start
- Security configuration
  - Security control - ELS, GDET0/1
  - Security boot
  - Security attestation
- Reset control
  - Monitors and release resets to individual peripherals
- Device ID register

17.3 Signals

Table 264. SYSCON pin description

Function	Type	Pin	Description
CLKOUT	O	PIO0_6, PIO2_2, PIO3_6	CLKOUT clock output. Refer to the Pinouts information.

17.4 Memory map and register definition

This section includes the SYSCON module memory map and detailed descriptions of all registers.

17.4.1 SYSCON register descriptions

17.4.1.1 SYSCON memory map

SYSCON0 base address: 4000_0000h

Offset	Register	Width (In bits)	Access	Reset value
10h	AHB Matrix Priority Control (AHBMATPRIO)	32	RW	See section
38h	Secure CPU0 System Tick Calibration (CPU0STCKCAL)	32	RW	See section
3Ch	Non-Secure CPU0 System Tick Calibration (CPU0NSTCKCAL)	32	RW	See section
40h	System tick calibration for CPU1 (CPU1STCKCAL)	32	RW	See section
48h	NMI Source Select (NMISRC)	32	RW	See section
100h	Peripheral Reset Control 0 (PRESETCTRL0)	32	RW	See section
104h	Peripheral Reset Control 1 (PRESETCTRL1)	32	RW	See section
108h	Peripheral Reset Control 2 (PRESETCTRL2)	32	RW	0000_0000h
10Ch	Peripheral Reset Control 3 (PRESETCTRL3)	32	RW	0000_0000h
120h - 12Ch	Peripheral Reset Control Set (PRESETCTRLSET0 - PRESETCTRLSET3)	32	W	0000_0000h
140h - 14Ch	Peripheral Reset Control Clear (PRESETCTRLCLR0 - PRESETCTRLCLR3)	32	W	0000_0000h
200h	AHB Clock Control 0 (AHBCLKCTRL0)	32	RW	0000_0603h
204h	AHB Clock Control 1 (AHBCLKCTRL1)	32	RW	0000_0000h
208h	AHB Clock Control 2 (AHBCLKCTRL2)	32	RW	0000_0000h
20Ch	AHB Clock Control 3 (AHBCLKCTRL3)	32	RW	0000_0000h
220h - 22Ch	AHB Clock Control Set (AHBCLKCTRLSET0 - AHBCLKCTRLSET3)	32	W	0000_0000h
240h - 24Ch	AHB Clock Control Clear (AHBCLKCTRLCLR0 - AHBCLKCTRLCLR3)	32	W	0000_0000h
260h	CPU0 System Tick Timer Source Select (SYSTICKCLKSEL0)	32	RW	0000_0007h
264h	CPU1 System Tick Timer Source Select (SYSTICKCLKSEL1)	32	RW	See section
268h	Trace Clock Source Select (TRACECLKSEL)	32	RW	See section
26Ch - 27Ch	CTIMER Clock Source Select (CTIMERCLKSEL0 - CTIMERCLKSEL4)	32	RW	See section
288h	CLKOUT Clock Source Select (CLKOUTSEL)	32	RW	See section
2A4h	ADC0 Clock Source Select (ADC0CLKSEL)	32	RW	See section
2A8h	USB-FS Clock Source Select (USB0CLKSEL)	32	RW	See section
2B0h - 2D4h	LP_FLEXCOMM Clock Source Select for Fractional Rate Divider (FCCLKSEL0 - FCCLKSEL9)	32	RW	See section
2F0h	SCTimer/PWM Clock Source Select (SCTCLKSEL)	32	RW	See section
300h	CPU0 System Tick Timer Divider (SYSTICKCLKDIV0)	32	RW	4000_0000h
304h	CPU1 System Tick Timer Divider (SYSTICKCLKDIV1)	32	RW	4000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
308h	TRACE Clock Divider (TRACECLKDIV)	32	RW	4000_0000h
350h	TSI Function Clock Source Select (TSICLKSEL)	32	RW	See section
360h	SINC FILTER Function Clock Source Select (SINCFILTCLKSEL)	32	RW	See section
378h	SLOW_CLK Clock Divider (SLOWCLKDIV)	32	RW	See section
37Ch	TSI Function Clock Divider (TSICLKDIV)	32	RW	4000_0000h
380h	System Clock Divider (AHBCLKDIV)	32	RW	0000_0000h
384h	CLKOUT Clock Divider (CLKOUTDIV)	32	RW	4000_0000h
388h	FRO_HF_DIV Clock Divider (FROHFDIV)	32	RW	4000_0000h
38Ch	WDT0 Clock Divider (WDT0CLKDIV)	32	RW	See section
394h	ADC0 Clock Divider (ADC0CLKDIV)	32	RW	See section
398h	USB-FS Clock Divider (USB0CLKDIV)	32	RW	4000_0000h
3B4h	SCT/PWM Clock Divider (SCTCLKDIV)	32	RW	4000_0000h
3C4h	PLL Clock Divider (PLLCLKDIV)	32	RW	4000_0000h
3D0h - 3E0h	CTimer Clock Divider (CTIMER0CLKDIV - CTIMER4CLKDIV)	32	RW	4000_0000h
3E4h	PLL1 Clock 0 Divider (PLL1CLK0DIV)	32	RW	4000_0000h
3E8h	PLL1 Clock 1 Divider (PLL1CLK1DIV)	32	RW	4000_0000h
3FCh	Clock Configuration Unlock (CLKUNLOCK)	32	RW	0000_0000h
400h	NVM Control (NVM_CTRL)	32	RW	0002_0410h
404h	ROM Wait State (ROMCR)	32	RW	0000_0000h
414h	SmartDMA Interrupt Hijack (SmartDMAINT)	32	RW	See section
464h	ADC1 Clock Source Select (ADC1CLKSEL)	32	RW	See section
468h	ADC1 Clock Divider (ADC1CLKDIV)	32	RW	See section
490h	DAC0 Functional Clock Selection (DAC0CLKSEL)	32	RW	0000_0007h
494h	DAC0 functional clock divider (DAC0CLKDIV)	32	RW	4000_0000h
498h	DAC1 Functional Clock Selection (DAC1CLKSEL)	32	RW	0000_0007h
49Ch	DAC1 functional clock divider (DAC1CLKDIV)	32	RW	4000_0000h
4A0h	DAC2 Functional Clock Selection (DAC2CLKSEL)	32	RW	0000_0007h
4A4h	DAC2 functional clock divider (DAC2CLKDIV)	32	RW	4000_0000h
4A8h	FlexSPI Clock Selection (FlexSPICLKSEL)	32	RW	0000_000Fh
4ACh	FlexSPI Clock Divider (FlexSPICLKDIV)	32	RW	4000_0000h
52Ch	PLL Clock Divider Clock Selection (PLLCLKDIVSEL)	32	RW	0000_0007h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
530h	I3C0 Functional Clock Selection (I3C0FCLKSEL)	32	RW	0000_0007h
534h	I3C0 FCLK_STC Clock Selection (I3C0FCLKSTCSEL)	32	RW	0000_0007h
538h	I3C0 FCLK_STC Clock Divider (I3C0FCLKSTCDIV)	32	RW	4000_0000h
53Ch	I3C0 FCLK Slow Clock Divider (I3C0FCLKSDIV)	32	RW	4000_0000h
540h	I3C0 Functional Clock FCLK Divider (I3C0FCLKDIV)	32	RW	4000_0000h
544h	I3C0 FCLK Slow Selection (I3C0FCLKSSEL)	32	RW	0000_0000h
548h	MICFIL Clock Selection (MICFILFCLKSEL)	32	RW	0000_000Fh
54Ch	MICFIL Clock Division (MICFILFCLKDIV)	32	RW	4000_0000h
558h	uSDHC Clock Selection (uSDHCCLKSEL)	32	RW	0000_0007h
55Ch	uSDHC Function Clock Divider (uSDHCCLKDIV)	32	RW	4000_0000h
560h	FLEXIO Clock Selection (FLEXIOCLKSEL)	32	RW	0000_0007h
564h	FLEXIO Function Clock Divider (FLEXIOCLKDIV)	32	RW	4000_0000h
5A0h	FLEXCAN0 Clock Selection (FLEXCAN0CLKSEL)	32	RW	0000_0007h
5A4h	FLEXCAN0 Function Clock Divider (FLEXCAN0CLKDIV)	32	RW	0000_0000h
5A8h	FLEXCAN1 Clock Selection (FLEXCAN1CLKSEL)	32	RW	0000_0007h
5ACh	FLEXCAN1 Function Clock Divider (FLEXCAN1CLKDIV)	32	RW	4000_0000h
5B0h	Ethernet RMII Clock Selection (ENETRMIICLKSEL)	32	RW	0000_0007h
5B4h	Ethernet RMII Function Clock Divider (ENETRMIICLKDIV)	32	RW	4000_0000h
5B8h	Ethernet PTP REF Clock Selection (ENETPTPREFCLKSEL)	32	RW	0000_0007h
5BCh	Ethernet PTP REF Function Clock Divider (ENETPTPREFCLKDIV)	32	RW	4000_0000h
5C0h	Ethernet PHY Interface Select (ENET_PHY_INTF_SEL)	32	RW	0000_0000h
5C4h	Sideband Flow Control (ENET_SBD_FLOW_CTRL)	32	RW	0000_0000h
5D4h	EWM0 Clock Selection (EWM0CLKSEL)	32	RW	0000_0001h
5D8h	WDT1 Clock Selection (WDT1CLKSEL)	32	RW	0000_0003h
5DCh	WDT1 Function Clock Divider (WDT1CLKDIV)	32	RW	4000_0000h
5E0h	OSTIMER Clock Selection (OSTIMERCLKSEL)	32	RW	0000_0003h
5F0h	CMP0 Function Clock Selection (CMP0FCLKSEL)	32	RW	0000_0007h
5F4h	CMP0 Function Clock Divider (CMP0FCLKDIV)	32	RW	4000_0000h
5F8h	CMP0 Round Robin Clock Selection (CMP0RRCLKSEL)	32	RW	0000_0007h
5FCh	CMP0 Round Robin Clock Divider (CMP0RRCLKDIV)	32	RW	4000_0000h
600h	CMP1 Function Clock Selection (CMP1FCLKSEL)	32	RW	0000_0007h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
604h	CMP1 Function Clock Divider (CMP1FCLKDIV)	32	RW	4000_0000h
608h	CMP1 Round Robin Clock Source Select (CMP1RRCLKSEL)	32	RW	0000_0007h
60Ch	CMP1 Round Robin Clock Division (CMP1RRCLKDIV)	32	RW	4000_0000h
610h	CMP2 Function Clock Source Select (CMP2FCLKSEL)	32	RW	0000_0007h
614h	CMP2 Function Clock Division (CMP2FCLKDIV)	32	RW	4000_0000h
618h	CMP2 Round Robin Clock Source Select (CMP2RRCLKSEL)	32	RW	0000_0007h
61Ch	CMP2 Round Robin Clock Division (CMP2RRCLKDIV)	32	RW	4000_0000h
800h	CPU Control for Multiple Processors (CPUCTRL)	32	RW	0000_0028h
804h	Coprocessor Boot Address (CPBOOT)	32	RW	0000_0000h
80Ch	CPU Status (CPUSTAT)	32	R	See section
824h	LPCAC Control (LPCAC_CTRL)	32	RW	0000_0031h
850h - 874h	LP_FLEXCOMM Clock Divider (FLEXCOMM0CLKDIV - FLEXCOMM9CLKDIV)	32	RW	4000_0000h
880h	SAI0 Function Clock Source Select (SAI0CLKSEL)	32	RW	0000_0007h
884h	SAI1 Function Clock Source Select (SAI1CLKSEL)	32	RW	0000_0007h
888h	SAI0 Function Clock Division (SAI0CLKDIV)	32	RW	4000_0000h
88Ch	SAI1 Function Clock Division (SAI1CLKDIV)	32	RW	4000_0000h
890h	EMVSIM0 Clock Source Select (EMVSIM0CLKSEL)	32	RW	0000_0007h
894h	EMVSIM1 Clock Source Select (EMVSIM1CLKSEL)	32	RW	0000_0007h
898h	EMVSIM0 Function Clock Division (EMVSIM0CLKDIV)	32	RW	4000_0000h
89Ch	EMVSIM1 Function Clock Division (EMVSIM1CLKDIV)	32	RW	4000_0000h
A18h	Clock Control (CLOCK_CTRL)	32	RW	See section
B30h	I3C1 Functional Clock Selection (I3C1FCLKSEL)	32	RW	0000_0007h
B34h	Selects the I3C1 Time Control clock (I3C1FCLKSTCSEL)	32	RW	0000_0007h
B38h	I3C1 FCLK_STC Clock Divider (I3C1FCLKSTCDIV)	32	RW	4000_0000h
B3Ch	I3C1 FCLK Slow clock Divider (I3C1FCLKSDIV)	32	RW	4000_0000h
B40h	I3C1 Functional Clock FCLK Divider (I3C1FCLKDIV)	32	RW	4000_0000h
B44h	I3C1 FCLK Slow Selection (I3C1FCLKSSEL)	32	RW	0000_0000h
B50h	ETB Counter Status Register (ETB_STATUS)	32	RW	0000_0000h
B54h	ETB Counter Control Register (ETB_COUNTER_CTRL)	32	RW	0000_0000h
B58h	ETB Counter Reload Register (ETB_COUNTER_RELOAD)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
B5Ch	ETB Counter Value Register (ETB_COUNTER_VALUE)	32	R	0000_0000h
B60h	Gray to Binary Converter Gray code_gray[31:0] (GRAY_CODE_LSB)	32	RW	0000_0000h
B64h	Gray to Binary Converter Gray code_gray[41:32] (GRAY_CODE_MSB)	32	RW	0000_0000h
B68h	Gray to Binary Converter Binary Code [31:0] (BINARY_CODE_LSB)	32	R	0000_0000h
B6Ch	Gray to Binary Converter Binary Code [41:32] (BINARY_CODE_MSB)	32	R	0000_0000h
E04h	Control Automatic Clock Gating (AUTOCLKGATEOVERRIDE)	32	RW	0000_FFFFh
E2Ch	Control Automatic Clock Gating C (AUTOCLKGATEOVERRIDEC)	32	RW	0000_0000h
E38h	PWM0 Submodule Control (PWM0SUBCTL)	32	RW	0000_0000h
E3Ch	PWM1 Submodule Control (PWM1SUBCTL)	32	RW	0000_0000h
E40h	CTIMER Global Start Enable (CTIMERGLOBALSTARTEN)	32	RW	0000_0000h
E44h	RAM ECC Enable Control (ECC_ENABLE_CTRL)	32	RW	0000_0003h
FF0h	JTAG Chip ID (JTAG_ID)	32	R	See section
FF4h	Device Type (DEVICE_TYPE)	32	R	See section
FF8h	Device ID (DEVICE_ID0)	32	R	See section
FFCh	Chip Revision ID and Number (DIEID)	32	R	See section

17.4.1.2 AHB Matrix Priority Control (AHBMATPRIO)

Offset

Register	Offset
AHBMATPRIO	10h

Function

The Multilayer AHB Matrix arbitrates between masters, when they attempt to access the same matrix slave port at the same time. The priority values are 3 = highest, 0 = lowest. When the priority is the same, the master with the lower master number is given priority.

NOTE

Be careful when modifying this register as improper settings can seriously degrade the performance.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		PRI_USDHC		PRI_USB_HS		PRI_USB_FS_ENET		PRI_NPU_D		PRI_COOLFLUX_Y_...		PRI_COOLFLUX_X		PRI_COOLFLUX_I	
Reset	u	u	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRI_NPU_PQ		Reserved		DMA1		DMA0		PRI_CPU1_CBUS_S...		PRI_CPU1_SBUS_S...		PRI_CPU0_SBUS		PRI_CPU0_CBUS	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29-28 PRI_USDHC	USDHC bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
27-26 PRI_USB_HS	USB-HS bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
25-24 PRI_USB_FS_ENET	USB-FS and ENET bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
23-22 PRI_NPU_D	NPU D bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3

Table continues on the next page...

Table continued from the previous page...

Field	Function
21-20 PRI_COOLFLUX_Y_ESPI	CoolFlux Y bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
19-18 PRI_COOLFLUX_X	CoolFlux X bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
17-16 PRI_COOLFLUX_I	CoolFlux I bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
15-14 PRI_NPU_PQ	NPU O bus and Powerquad bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
13-12 —	Reserved
11-10 DMA1	DMA1 controller bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
9-8 DMA0	DMA0 controller bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-6 PRI_CPU1_CBUS_SmartDMA_I	CPU1 C-AHB/SmartDMA-I bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
5-4 PRI_CPU1_SBUS_SmartDMA_D	CPU1 S-AHB/SmartDMA-D bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
3-2 PRI_CPU0_SBUS	CPU0 S-AHB bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3
1-0 PRI_CPU0_CBUS	CPU0 C-AHB bus master priority level 00b - level 0 01b - level 1 10b - level 2 11b - level 3

17.4.1.3 Secure CPU0 System Tick Calibration (CPU0STCKCAL)

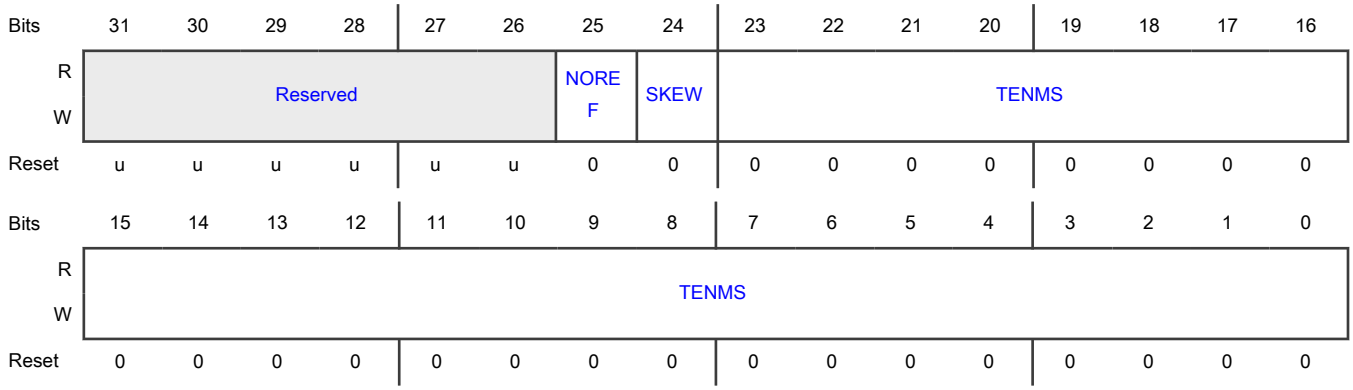
Offset

Register	Offset
CPU0STCKCAL	38h

Function

The CPU0STCKCAL register allows software to set up a default value for the SYST_CALIB register (refer to Arm documentation) in the System Tick Timer of secure part of the CPU0.

Diagram



Fields

Field	Function
31-26 —	Reserved
25 NOREF	Whether the device provides a reference clock to the processor. 0b - Reference clock is provided 1b - No reference clock is provided
24 SKEW	Whether the TENMS value is exact. 0b - TENMS value is exact 1b - TENMS value is not exact or not given
23-0 TENMS	Reload value for 10 ms (100 Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.

17.4.1.4 Non-Secure CPU0 System Tick Calibration (CPU0NSTCKCAL)

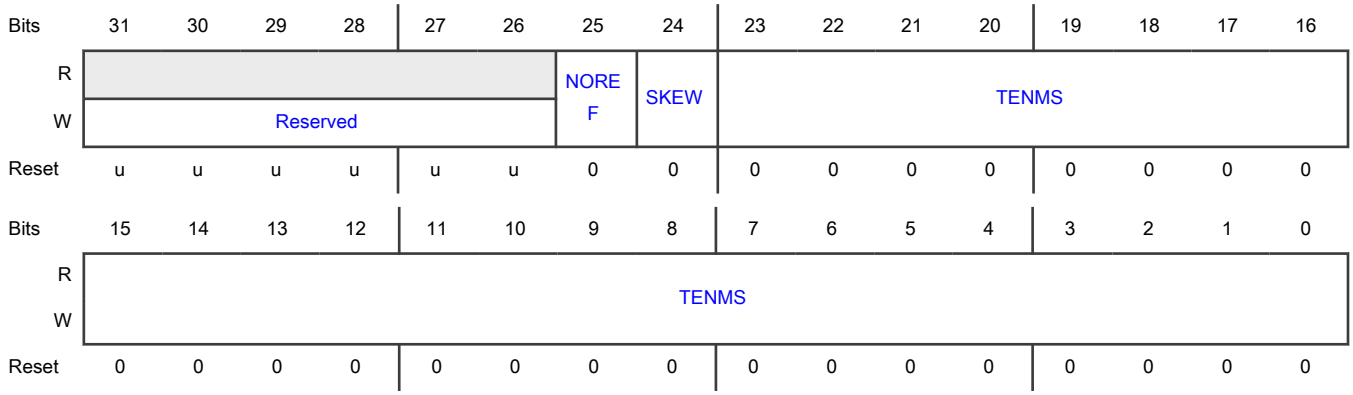
Offset

Register	Offset
CPU0NSTCKCAL	3Ch

Function

The CPU0NSTCKCAL register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of non-secure part of the CPU0.

Diagram



Fields

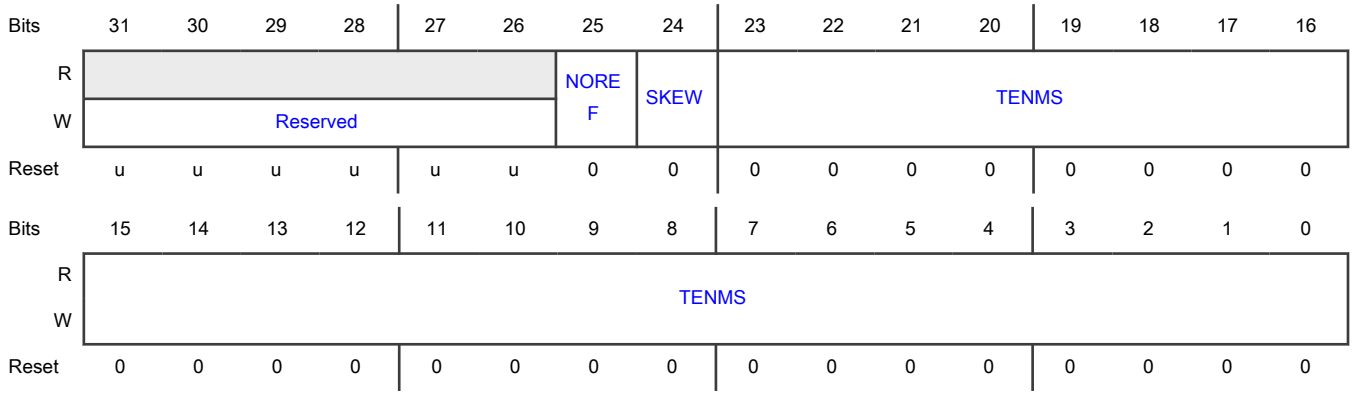
Field	Function
31-26 —	Reserved
25 NOREF	Indicates whether the device provides a reference clock to the processor. 0b - Reference clock is provided 1b - No reference clock is provided
24 SKEW	Indicates whether the TENMS value is exact. 0b - TENMS value is exact 1b - TENMS value is not exact or not given
23-0 TENMS	Reload value for 10 ms (100 Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.

17.4.1.5 System tick calibration for CPU1 (CPU1STCKCAL)

Offset

Register	Offset
CPU1STCKCAL	40h

Diagram



Fields

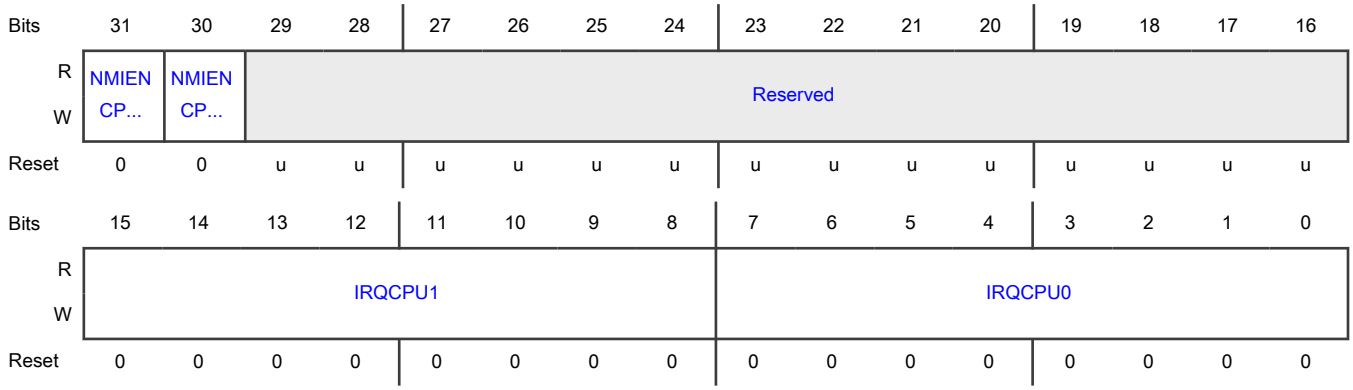
Field	Function
31-26 —	Reserved
25 NOREF	Indicates whether the device provides a reference clock to the processor. 0b - Reference clock is provided 1b - No reference clock is provided
24 SKEW	Indicates whether the TENMS value is exact. 0b - TENMS value is exact 1b - TENMS value is not exact or not given
23-0 TENMS	Reload value for 10 ms (100 Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.

17.4.1.6 NMI Source Select (NMISRC)

Offset

Register	Offset
NMISRC	48h

Diagram



Fields

Field	Function
31 NMIENCPU0	Enables the Non-Maskable Interrupt (NMI) source selected by IRQCPU0. 0b - Disable. 1b - Enable.
30 NMIENCPU1	Enables the Non-Maskable Interrupt (NMI) source selected by IRQCPU1. 0b - Disable. 1b - Enable.
29-16 —	Reserved
15-8 IRQCPU1	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for CPU1, if enabled by NMIENCPU1.
7-0 IRQCPU0	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for CPU0, if enabled by NMIENCPU0.

17.4.1.7 Peripheral Reset Control 0 (PRESETCTRL0)

Offset

Register	Offset
PRESETCTRL0	100h

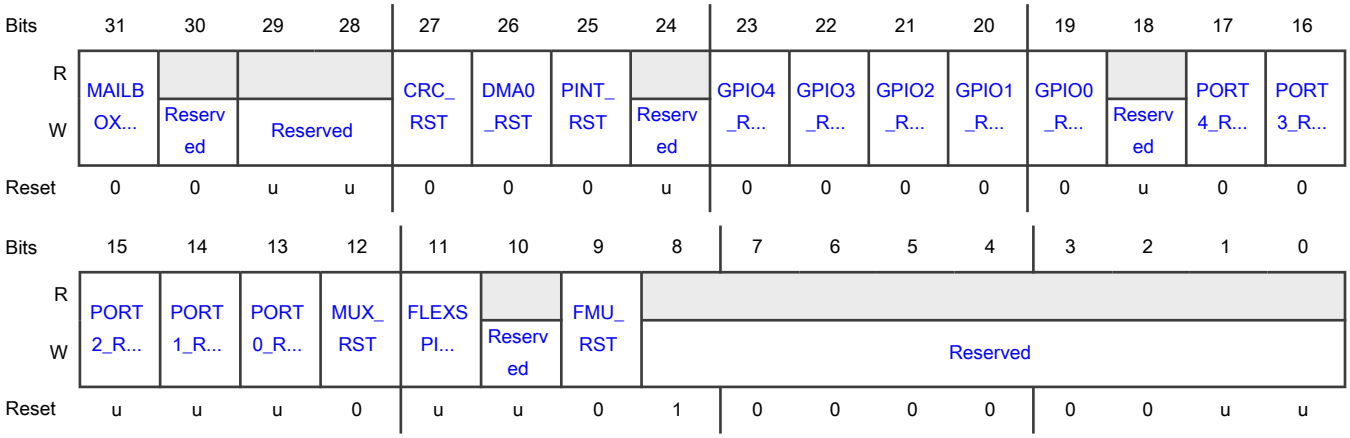
Function

The PRESETCTRL0 register allows software to reset specific peripherals. Writing a 0 to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a 1 asserts the reset.

NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram



Fields

Field	Function
31 MAILBOX_RST	Inter-CPU communication Mailbox reset control 0b - Block is not reset 1b - Block is reset
30 —	Reserved Read value is undefined, only zero should be written.
29-28 —	Reserved
27 CRC_RST	CRC reset control 0b - Block is not reset 1b - Block is reset
26 DMA0_RST	DMA0 reset control 0b - Block is not reset 1b - Block is reset
25 PINT_RST	PINT reset control 0b - Block is not reset 1b - Block is reset
24 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 GPIO4_RST	GPIO4 reset control 0b - Block is not reset 1b - Block is reset
22 GPIO3_RST	GPIO3 reset control 0b - Block is not reset 1b - Block is reset
21 GPIO2_RST	GPIO2 reset control 0b - Block is not reset 1b - Block is reset
20 GPIO1_RST	GPIO1 reset control 0b - Block is not reset 1b - Block is reset
19 GPIO0_RST	GPIO0 reset control 0b - Block is not reset 1b - Block is reset
18 —	Reserved
17 PORT4_RST	PORT4 reset control 0b - Block is not reset 1b - Block is reset
16 PORT3_RST	PORT3 reset control 0b - Block is not reset 1b - Block is reset
15 PORT2_RST	PORT2 reset control 0b - Block is not reset 1b - Block is reset
14 PORT1_RST	PORT1 reset control 0b - Block is not reset 1b - Block is reset
13 PORT0_RST	PORT0 controller reset control 0b - Block is not reset 1b - Block is reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 MUX_RST	INPUTMUX reset control 0b - Block is not reset 1b - Block is reset
11 FLEXSPI_RST	FlexSPI reset control 0b - Block is not reset 1b - Block is reset
10 —	Reserved
9 FMU_RST	Flash management unit reset control 0b - Block is not reset 1b - Block is reset
8-0 —	Reserved

17.4.1.8 Peripheral Reset Control 1 (PRESETCTRL1)

Offset

Register	Offset
PRESETCTRL1	104h

Function

The PRESETCTRL1 register allows software to reset specific peripherals. Writing a 0 to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a 1 asserts the reset.

NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Smart DM...				TIMER 1_...	TIMER 0_...	USB0_ FS...	USB0_ FS...	Reserv ed	TIMER 2_...	MICFI L_...	FC9_ RST	FC8_ RST	FC7_ RST	FC6_ RST	FC5_ RST
W		Reserv ed	Reserv ed	Reserv ed												
Reset	0	u	0	u	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FC4_ RST	FC3_ RST	FC2_ RST	FC1_ RST	FC0_ RST	UTICK _R...	EVSIM 1_...	EVSIM 0_...		RTC_ RST	DAC0_ RST	ADC1_ RST	ADC0_ RST	SCT_ RST	OSTIM ER...	MRT_ RST
W									Reserv ed							
Reset	0	0	0	0	0	0	0	0	u	u	0	0	0	0	0	0

Fields

Field	Function
31 SmartDMA_RST	SmartDMA reset control 0b - Block is not reset 1b - Block is reset
30 —	Reserved
29 —	Reserved
28 —	Reserved
27 TIMER1_RST	CTIMER1 reset control 0b - Block is not reset 1b - Block is reset
26 TIMER0_RST	CTIMER0 reset control 0b - Block is not reset 1b - Block is reset
25 USB0_FS_RST	USB FS reset control 0b - Block is not reset 1b - Block is reset
24 USB0_FS_DCD_RST	USB FS DCD reset control 0b - Block is not reset 1b - Block is reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 —	Reserved
22 TIMER2_RST	CTIMER2 reset control 0b - Block is not reset 1b - Block is reset
21 MICFIL_RST	MICFIL reset control 0b - Block is not reset 1b - Block is reset
20 FC9_RST	LP_FLEXCOMM9 reset control 0b - Block is not reset 1b - Block is reset
19 FC8_RST	LP_FLEXCOMM8 reset control 0b - Block is not reset 1b - Block is reset
18 FC7_RST	LP_FLEXCOMM7 reset control 0b - Block is not reset 1b - Block is reset
17 FC6_RST	LP_FLEXCOMM6 reset control 0b - Block is not reset 1b - Block is reset
16 FC5_RST	LP_FLEXCOMM5 reset control 0b - Block is not reset 1b - Block is reset
15 FC4_RST	LP_FLEXCOMM4 reset control 0b - Block is not reset 1b - Block is reset
14 FC3_RST	LP_FLEXCOMM3 reset control 0b - Block is not reset 1b - Block is reset
13 FC2_RST	LP_FLEXCOMM2 reset control 0b - Block is not reset 1b - Block is reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 FC1_RST	LP_FLEXCOMM1 reset control 0b - Block is not reset 1b - Block is reset
11 FC0_RST	LP_FLEXCOMM0 reset control 0b - Block is not reset 1b - Block is reset
10 UTICK_RST	UTICK reset control 0b - Block is not reset 1b - Block is reset
9 EVSIM1_RST	EVSIM1 reset control 0b - Block is not reset 1b - Block is reset
8 EVSIM0_RST	EVSIM0 reset control 0b - Block is not reset 1b - Block is reset
7 —	Reserved
6 RTC_RST	RTC reset control RTC and Sub-second counter reset control **NOTE** To reset RTC correctly, you must write 1 to RTC.CTRL[SWR] when writing 1 to RTC_RST, and write 0 to RTC.CTRL[SWR] after writing 0 to RTC_RST. 0b - Block is not reset 1b - Block is reset
5 DAC0_RST	DAC0 reset control 0b - Block is not reset 1b - Block is reset
4 ADC1_RST	ADC1 reset control 0b - Block is not reset 1b - Block is reset
3 ADC0_RST	ADC0 reset control

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Block is not reset 1b - Block is reset
2 SCT_RST	SCT reset control 0b - Block is not reset 1b - Block is reset
1 OSTIMER_RST	OS Event Timer reset control 0b - Block is not reset 1b - Block is reset
0 MRT_RST	MRT reset control 0b - Block is not reset 1b - Block is reset

17.4.1.9 Peripheral Reset Control 2 (PRESETCTRL2)

Offset

Register	Offset
PRESETCTRL2	108h

Function

The PRESETCTRL2 register allows software to reset specific peripherals. Writing a 0 to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a 1 asserts the reset.

NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R										TIMER4_...	TIMER3_...	PLU_RST	PQ_RST		USB_HS_...	USB_HS_...
W	Reserved	Reserved	Reserved					Reserved	Reserved					Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEXCAN...	FLEXCAN...	Reserved					FREQME_...		SAI1_RST	SAI0_RST	FLEXIO_...	USDHC_R...	ENET_RST	DMA1_RST	Reserved
W				Reserved					Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29-25 —	Reserved
24 —	Reserved
23 —	Reserved
22 TIMER4_RST	CTIMER4 reset control 0b - Block is not reset 1b - Block is reset
21 TIMER3_RST	CTIMER3 reset control 0b - Block is not reset 1b - Block is reset
20 PLU_RST	PLU reset control 0b - Block is not reset 1b - Block is reset
19 PQ_RST	PowerQuad reset control 0b - Block is not reset 1b - Block is reset
18 —	Reserved
17 USB_HS_PHY_RST	USB HS PHY reset control 0b - Block is not reset 1b - Block is reset
16 USB_HS_RST	USB HS reset control 0b - Block is not reset 1b - Block is reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 FLEXCAN1_RST	CAN1 reset control 0b - Block is not reset 1b - Block is reset
14 FLEXCAN0_RST	CAN0 reset control 0b - Block is not reset 1b - Block is reset
13 —	Reserved
12-9 —	Reserved
8 FREQME_RST	FREQME reset control 0b - Block is not reset 1b - Block is reset
7 —	Reserved Should keep default value.
6 SAI1_RST	SAI1 reset control 0b - Block is not reset 1b - Block is reset
5 SAI0_RST	SAI0 reset control 0b - Block is not reset 1b - Block is reset
4 FLEXIO_RST	FLEXIO reset control 0b - Block is not reset 1b - Block is reset
3 USDHC_RST	uSDHC reset control 0b - Block is not reset 1b - Block is reset
2 ENET_RST	Ethernet reset control 0b - Block is not reset 1b - Block is reset
1	DMA1 reset control

Table continues on the next page...

Table continued from the previous page...

Field	Function
DMA1_RST	0b - Block is not reset 1b - Block is reset
0 —	Reserved

17.4.1.10 Peripheral Reset Control 3 (PRESETCTRL3)

Offset

Register	Offset
PRESETCTRL3	10Ch

Function

The PRESETCTRL3 register allows software to reset specific peripherals. Writing a 0 to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a 1 asserts the reset.

NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				SEMA42_...	Reserved		EIM_RST	EWM_RST	TSI_RST	NPU_RST	COOLFLU...	VREF_RST	CMP2_RST	Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAMP2_...	OPAMP1_...	OPAMP0_...	DAC2_RST	DAC1_RST	Reserved	Reserved	AOI0_RST	PWM1_RST	PWM0_RST	QDC1_RST	QDC0_RST	COOLFLU...	SINC_RST	I3C1_RST	I3C0_RST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27	Semaphore reset control

Table continues on the next page...

Table continued from the previous page...

Field	Function
SEMA42_RST	0b - Block is not reset 1b - Block is reset
26-25 —	Reserved
24 EIM_RST	EIM reset control 0b - Block is not reset 1b - Block is reset
23 EWM_RST	EWM reset control 0b - Block is not reset 1b - Block is reset
22 TSI_RST	TSI reset control 0b - Block is not reset 1b - Block is reset
21 NPU_RST	NPU reset control 0b - Block is not reset 1b - Block is reset
20 COOLFLUX_APB_RST	CoolFlux APB reset control 0b - Block is not reset 1b - Block is reset
19 VREF_RST	VREF reset control 0b - Block is not reset 1b - Block is reset
18 CMP2_RST	CMP2 reset control 0b - Block is not reset 1b - Block is reset
17-16 —	Reserved
15 OPAMP2_RST	OPAMP2 reset control 0b - Block is not reset 1b - Block is reset
14	OPAMP1 reset control

Table continues on the next page...

Table continued from the previous page...

Field	Function
OPAMP1_RST	0b - Block is not reset 1b - Block is reset
13 OPAMP0_RST	OPAMP0 reset control 0b - Block is not reset 1b - Block is reset
12 DAC2_RST	DAC2 reset control 0b - Block is not reset 1b - Block is reset
11 DAC1_RST	DAC1 reset control 0b - Block is not reset 1b - Block is reset
10 —	Reserved Read value is undefined, only zero should be written.
9 —	Reserved Read value is undefined, only zero should be written.
8 AOI0_RST	AOI0 reset control 0b - Block is not reset 1b - Block is reset
7 PWM1_RST	PWM1 reset control 0b - Block is not reset 1b - Block is reset
6 PWM0_RST	PWM0 reset control 0b - Block is not reset 1b - Block is reset
5 QDC1_RST	QDC1 reset control 0b - Block is not reset 1b - Block is reset
4 QDC0_RST	QDC0 reset control 0b - Block is not reset 1b - Block is reset
3	CoolFlux reset control

Table continues on the next page...

Table continued from the previous page...

Field	Function
COOLFLUX_RST	0b - Block is not reset 1b - Block is reset
2 SINC_RST	SINC reset control 0b - Block is not reset 1b - Block is reset
1 I3C1_RST	I3C1 reset control 0b - Block is not reset 1b - Block is reset
0 I3C0_RST	I3C0 reset control 0b - Block is not reset 1b - Block is reset

17.4.1.11 Peripheral Reset Control Set (PRESETCTRLSET0 - PRESETCTRLSET3)

Offset

Register	Offset
PRESETCTRLSET0	120h
PRESETCTRLSET1	124h
PRESETCTRLSET2	128h
PRESETCTRLSET3	12Ch

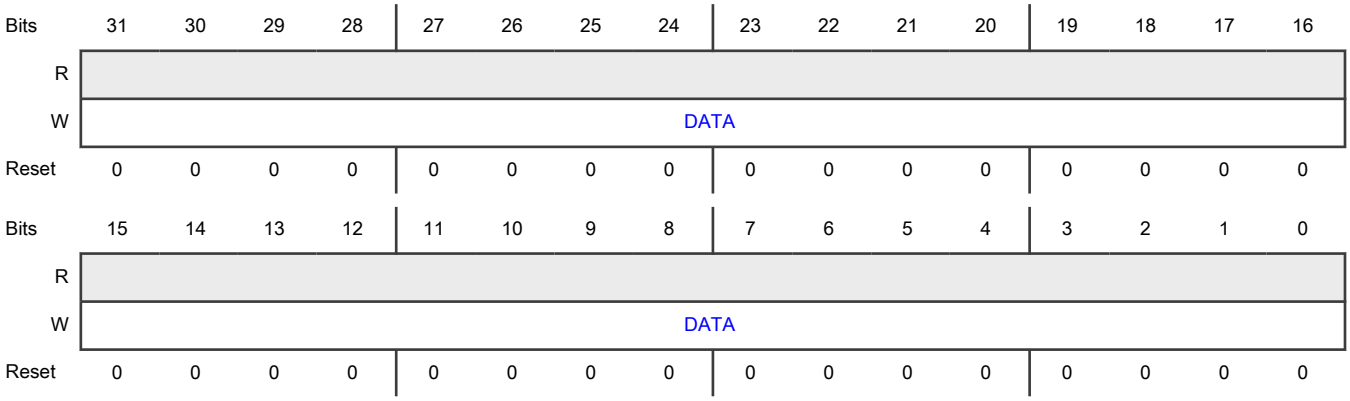
Function

Writing a 1 to a bit position in a write-only PRESETCTRLSETn register sets the corresponding position in PRESETCTRLn.

NOTE

To reset RTC correctly, you must write 1 to RTC.CTRL[SWR] when writing 1 to PRESETCTRL1[RTC_RST], and write 0 to RTC.CTRL[SWR] after writing 0 to PRESETCTRL1[RTC_RST].

Diagram



Fields

Field	Function
31-0 DATA	Data array value, refer to corresponding position in PRESETCTRLn.

17.4.1.12 Peripheral Reset Control Clear (PRESETCTRLCLR0 - PRESETCTRLCLR3)

Offset

Register	Offset
PRESETCTRLCLR0	140h
PRESETCTRLCLR1	144h
PRESETCTRLCLR2	148h
PRESETCTRLCLR3	14Ch

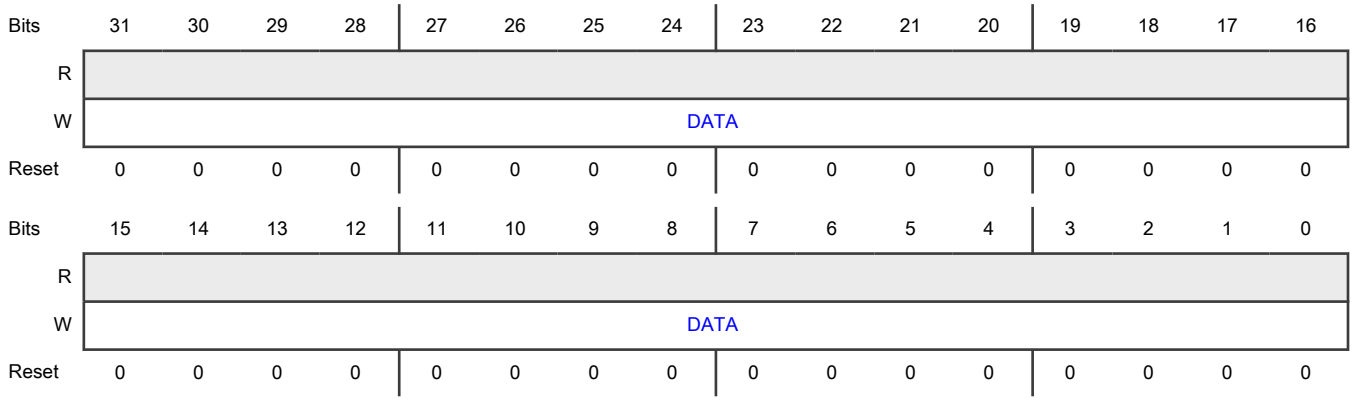
Function

Writing a 1 to a bit position in a write-only PRESETCTRLCLRn register clears the corresponding position in PRESETCTRLn.

NOTE

To reset RTC correctly, you must write 1 to RTC.CTRL[SWR] when writing 1 to PRESETCTRL1[RTC_RST], and write 0 to RTC.CTRL[SWR] after writing 0 to PRESETCTRL1[RTC_RST].

Diagram



Fields

Field	Function
31-0 DATA	Data array value, refer to corresponding position in PRESETCTRLn.

17.4.1.13 AHB Clock Control 0 (AHBCLKCTRL0)

Offset

Register	Offset
AHBCLKCTRL0	200h

Function

The AHBCLKCTRLn registers enable the clocks of the individual modules.

NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAILBOX	Reserved	WWDT1	WWDT0	CRC	DMA0	PINT	Reserved	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	Reserved	PORT4	PORT3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PORT2	PORT1	PORT0	MUX	FLEXSPI	FMC	FMU	RAMH_CT...	RAMG_CT...	RAMF_CT...	RAME_CT...	RAMD_CT...	RAMC_CT...	RAMB_CT...	ROM	Reserved
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1

Fields

Field	Function
31 MAILBOX	Enables the clock for the Inter CPU communication Mailbox. 0b - Disables clock 1b - Enables clock
30 —	Reserved Read value is undefined, only zero should be written.
29 WWDT1	Enables the clock for WWDT1 0b - Disables clock 1b - Enables clock
28 WWDT0	Enables the clock for WWDT0 0b - Disables clock 1b - Enables clock
27 CRC	Enables the clock for CRC 0b - Disables clock 1b - Enables clock
26 DMA0	Enables the clock for DMA0 0b - Disables clock 1b - Enables clock
25 PINT	Enables the clock for PINT 0b - Disables clock 1b - Enables clock
24	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
23 GPIO4	Enables the clock for GPIO4 0b - Disables clock 1b - Enables clock
22 GPIO3	Enables the clock for GPIO3 0b - Disables clock 1b - Enables clock
21 GPIO2	Enables the clock for GPIO2 0b - Disables clock 1b - Enables clock
20 GPIO1	Enables the clock for GPIO1 0b - Disables clock 1b - Enables clock
19 GPIO0	Enables the clock for GPIO0 0b - Disables clock 1b - Enables clock
18 —	Reserved
17 PORT4	Enables the clock for PORT4 0b - Disables clock 1b - Enables clock
16 PORT3	Enables the clock for PORT3 0b - Disables clock 1b - Enables clock
15 PORT2	Enables the clock for PORT2 0b - Disables clock 1b - Enables clock
14 PORT1	Enables the clock for PORT1 0b - Disables clock 1b - Enables clock
13	Enables the clock for PORT0 controller

Table continues on the next page...

Table continued from the previous page...

Field	Function
PORT0	0b - Disables clock 1b - Enables clock
12 MUX	Enables the clock for INPUTMUX 0b - Disables clock 1b - Enables clock
11 FLEXSPI	Enables the clock for FlexSPI 0b - Disables clock 1b - Enables clock
10 FMC	Enables the clock for the Flash Memory Controller 0b - Disables clock 1b - Enables clock
9 FMU	Enables the clock for the Flash Management Unit 0b - Disables clock 1b - Enables clock
8 RAMH_CTRL	Enables the clock for the RAMH Controller 0b - Disables clock 1b - Enables clock
7 RAMG_CTRL	Enables the clock for the RAMG Controller 0b - Disables clock 1b - Enables clock
6 RAMF_CTRL	Enables the clock for the RAMF Controller 0b - Disables clock 1b - Enables clock
5 RAME_CTRL	Enables the clock for the RAME Controller 0b - Disables clock 1b - Enables clock
4 RAMD_CTRL	Enables the clock for the RAMD Controller 0b - Disables clock 1b - Enables clock
3 RAMC_CTRL	Enables the clock for the RAMC Controller 0b - Disables clock 1b - Enables clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 RAMB_CTRL	Enables the clock for the RAMB Controller 0b - Disables clock 1b - Enables clock
1 ROM	Enables the clock for the ROM 0b - Disables clock 1b - Enables clock
0 —	Reserved Read value is undefined, only zero should be written.

17.4.1.14 AHB Clock Control 1 (AHBCLKCTRL1)

Offset

Register	Offset
AHBCLKCTRL1	204h

Function

The AHBCLKCTRLn registers enable the clocks of the individual modules.

NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Smart DMA				TIMER 1	TIMER 0	USB0_ FS	USB0_ FS...	Reserv ed	TIMER 2	MICFI L	FC9	FC8	FC7	FC6	FC5
W		Reserv ed	Reserv ed	Reserv ed												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FC4	FC3	FC2	FC1	FC0	UTICK	EVSIM 1	EVSIM 0		RTC	DAC0	ADC1	ADC0	SCT	OSTIM ER	MRT
W									Reserv ed							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SmartDMA	Enables the clock for SmartDMA 0b - Disables clock 1b - Enables clock
30 —	Reserved
29 —	Reserved
28 —	Reserved
27 TIMER1	Enables the clock for CTIMER1 0b - Disables clock 1b - Enables clock
26 TIMER0	Enables the clock for CTIMER0 0b - Disables clock 1b - Enables clock
25 USB0_FS	Enables the clock for USB-FS 0b - Disables clock 1b - Enables clock
24 USB0_FS_DCD	Enables the clock for USB-FS DCD 0b - Disables clock 1b - Enables clock
23 —	Reserved
22 TIMER2	Enables the clock for CTIMER2 0b - Disables clock 1b - Enables clock
21 MICFIL	Enables the clock for MICFIL 0b - Disables clock 1b - Enables clock
20 FC9	Enables the clock for LP_FLEXCOMM9

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disables clock 1b - Enables clock
19 FC8	Enables the clock for LP_FLEXCOMM8 0b - Disables clock 1b - Enables clock
18 FC7	Enables the clock for LP_FLEXCOMM7 0b - Disables clock 1b - Enables clock
17 FC6	Enables the clock for LP_FLEXCOMM6 0b - Disables clock 1b - Enables clock
16 FC5	Enables the clock for LP_FLEXCOMM5 0b - Disables clock 1b - Enables clock
15 FC4	Enables the clock for LP_FLEXCOMM4 0b - Disables clock 1b - Enables clock
14 FC3	Enables the clock for LP_FLEXCOMM3 0b - Disables clock 1b - Enables clock
13 FC2	Enables the clock for LP_FLEXCOMM2 0b - Disables clock 1b - Enables clock
12 FC1	Enables the clock for LP_FLEXCOMM1 0b - Disables clock 1b - Enables clock
11 FC0	Enables the clock for LP_FLEXCOMM0 0b - Disables clock 1b - Enables clock
10 UTICK	Enables the clock for UTICK 0b - Disables clock 1b - Enables clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 EVSIM1	Enables the clock for EVSIM1 0b - Disables clock 1b - Enables clock
8 EVSIM0	Enables the clock for EVSIM0 0b - Disables clock 1b - Enables clock
7 —	Reserved
6 RTC	Enables the clock for RTC 0b - Disables clock 1b - Enables clock
5 DAC0	Enables the clock for DAC0 0b - Disables clock 1b - Enables clock
4 ADC1	Enables the clock for ADC1 0b - Disables clock 1b - Enables clock
3 ADC0	Enables the clock for ADC0 0b - Disables clock 1b - Enables clock
2 SCT	Enables the clock for SCT 0b - Disables clock 1b - Enables clock
1 OSTIMER	Enables the clock for the OS Event Timer 0b - Disables clock 1b - Enables clock
0 MRT	Enables the clock for MRT 0b - Disables clock 1b - Enables clock

17.4.1.15 AHB Clock Control 2 (AHBCLKCTRL2)

Offset

Register	Offset
AHBCLKCTRL2	208h

Function

The AHBCLKCTRLn registers enable the clocks of the individual modules.

NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved	Reserved	Reserved	Reserved		SCG	Reserved	Reserved		TIMER 4	TIMER 3	PLU_ LUT	PQ	Reserved	USB_ HS_...	USB_ HS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	FLEX CAN1	FLEX CAN0	Reserved	Reserved				FREQ ME	Reserved	SAI1	SAI0	FLEXI O	uSDH C	ENET	DMA1	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28-27 —	Reserved
26	Enables the clock for SCG

Table continues on the next page...

Table continued from the previous page...

Field	Function
SCG	0b - Disables clock 1b - Enables clock
25 —	Reserved Read value is undefined, only zero should be written.
24-23 —	Reserved
22 TIMER4	Enables the clock for CTIMER4 0b - Disables clock 1b - Enables clock
21 TIMER3	Enables the clock for CTIMER3 0b - Disables clock 1b - Enables clock
20 PLU_LUT	Enables the clock for PLU_LUT 0b - Disables clock 1b - Enables clock
19 PQ	Enables the clock for Powerquad 0b - Disables clock 1b - Enables clock
18 —	Reserved
17 USB_HS_PHY	Enables the clock for USB HS PHY 0b - Disables clock 1b - Enables clock
16 USB_HS	Enables the clock for USB HS 0b - Disables clock 1b - Enables clock
15 FLEXCAN1	Enables the clock for FLEXCAN1 0b - Disables clock 1b - Enables clock
14 FLEXCAN0	Enables the clock for FLEXCAN0

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disables clock 1b - Enables clock
13 —	Reserved
12-9 —	Reserved
8 FREQME	Enables the clock for the Frequency meter 0b - Disables clock 1b - Enables clock
7 —	Reserved Should keep default value.
6 SAI1	Enables the clock for SAI1 0b - Disables clock 1b - Enables clock
5 SAI0	Enables the clock for SAI0 0b - Disables clock 1b - Enables clock
4 FLEXIO	Enables the clock for FlexIO 0b - Disables clock 1b - Enable clock
3 uSDHC	Enables the clock for uSDHC 0b - Disables clock 1b - Enables clock
2 ENET	Enables the clock for Ethernet 0b - Disables clock 1b - Enables clock
1 DMA1	Enables the clock for DMA1 0b - Disables clock 1b - Enables clock
0 —	Reserved

17.4.1.16 AHB Clock Control 3 (AHBCLKCTRL3)

Offset

Register	Offset
AHBCLKCTRL3	20Ch

Function

The AHBCLKCTRLn registers enable the clocks of the individual modules.

NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				SEMA42	INTM	ERM	EIM	EWM	TSI	NPU	COOLFLU...	VREF	CMP2		
W															Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OPAMP2	OPAMP1	OPAMP0	DAC2	DAC1	Reserved	Reserved	EVTG	PWM1	PWM0	QDC1	QDC0	COOLFLUX	SINC	I3C1	I3C0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27 SEMA42	Enables the clock for Semaphore 0b - Disables clock 1b - Enables clock
26 INTM	Enables the clock for INTM 0b - Disables clock 1b - Enables clock
25 ERM	Enables the clock for ERM 0b - Disables clock 1b - Enables clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
24 EIM	Enables the clock for EIM 0b - Disables clock 1b - Enables clock
23 EWM	Enables the clock for EWM 0b - Disables clock 1b - Enables clock
22 TSI	Enables the clock for TSI 0b - Disables clock 1b - Enables clock
21 NPU	Enables the clock for NPU 0b - Disables clock 1b - Enables clock
20 COOLFLUX_APB	Enables the clock for CoolFlux APB 0b - Disables clock 1b - Enables clock (CoolFlux needs to be properly programmed before the clock enabled.)
19 VREF	Enables the clock for VREF 0b - Disables clock 1b - Enables clock
18 CMP2	Enables the clock for CMP2 0b - Disables clock 1b - Enables clock
17-16 —	Reserved
15 OPAMP2	Enables the clock for OPAMP2 0b - Disables clock 1b - Enables clock
14 OPAMP1	Enables the clock for OPAMP1 0b - Disables clock 1b - Enables clock
13 OPAMP0	Enables the clock for OPAMP0 0b - Disables clock 1b - Enables clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 DAC2	Enables the clock for DAC2 0b - Disables clock 1b - Enables clock
11 DAC1	Enables the clock for DAC1 0b - Disables clock 1b - Enables clock
10 —	Reserved Read value is undefined, only zero should be written.
9 —	Reserved Read value is undefined, only zero should be written.
8 EVTG	Enables the clock for EVTG 0b - Disables clock 1b - Enables clock
7 PWM1	Enables the clock for PWM1 0b - Disables clock 1b - Enables clock
6 PWM0	Enables the clock for PWM0 0b - Disables clock 1b - Enables clock
5 QDC1	Enables the clock for QDC1 0b - Disables clock 1b - Enables clock
4 QDC0	Enables the clock for QDC0 0b - Disables clock 1b - Enables clock
3 COOLFLUX	Enables the clock for CoolFlux 0b - Disables clock 1b - Enables clock
2 SINC	Enables the clock for SINC 0b - Disables clock 1b - Enables clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 I3C1	Enables the clock for I3C1 0b - Disables clock 1b - Enables clock
0 I3C0	Enables the clock for I3C0 0b - Disables clock 1b - Enables clock

17.4.1.17 AHB Clock Control Set (AHBCLKCTRLSET0 - AHBCLKCTRLSET3)

Offset

Register	Offset
AHBCLKCTRLSET0	220h
AHBCLKCTRLSET1	224h
AHBCLKCTRLSET2	228h
AHBCLKCTRLSET3	22Ch

Function

Writing a 1 to a bit position in a write-only AHBCLKCTRLSETn register sets the corresponding position in AHBCLKCTRLn.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DATA	Data array value

17.4.1.18 AHB Clock Control Clear (AHBCLKCTRLCLR0 - AHBCLKCTRLCLR3)

Offset

Register	Offset
AHBCLKCTRLCLR0	240h
AHBCLKCTRLCLR1	244h
AHBCLKCTRLCLR2	248h
AHBCLKCTRLCLR3	24Ch

Function
Writing a 1 to a bit position in a write-only AHBCLKCTRLCLRn register clears the corresponding position in AHBCLKCTRLn.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

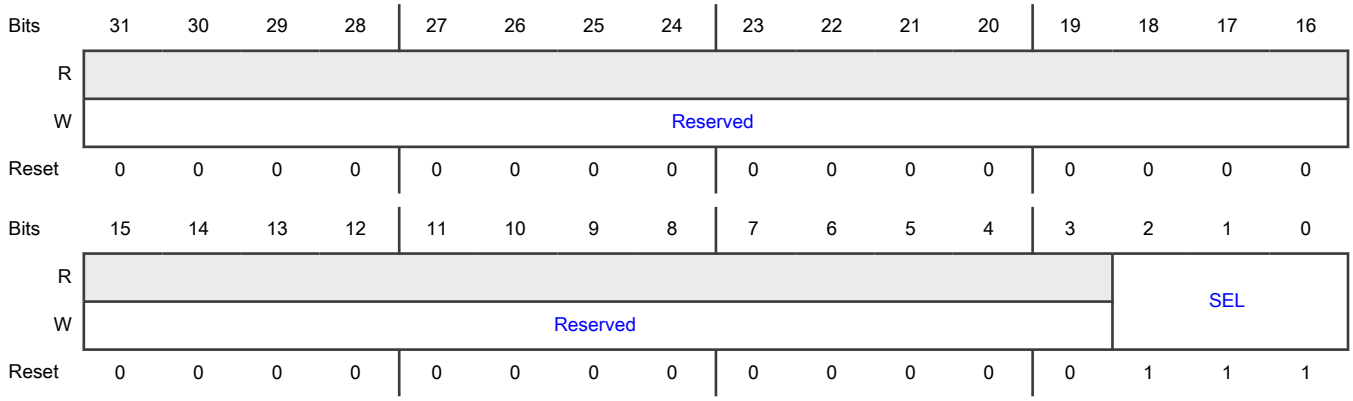
Field	Function
31-0 DATA	Data array value

17.4.1.19 CPU0 System Tick Timer Source Select (SYSTICKCLKSEL0)

Offset

Register	Offset
SYSTICKCLKSEL0	260h

Diagram



Fields

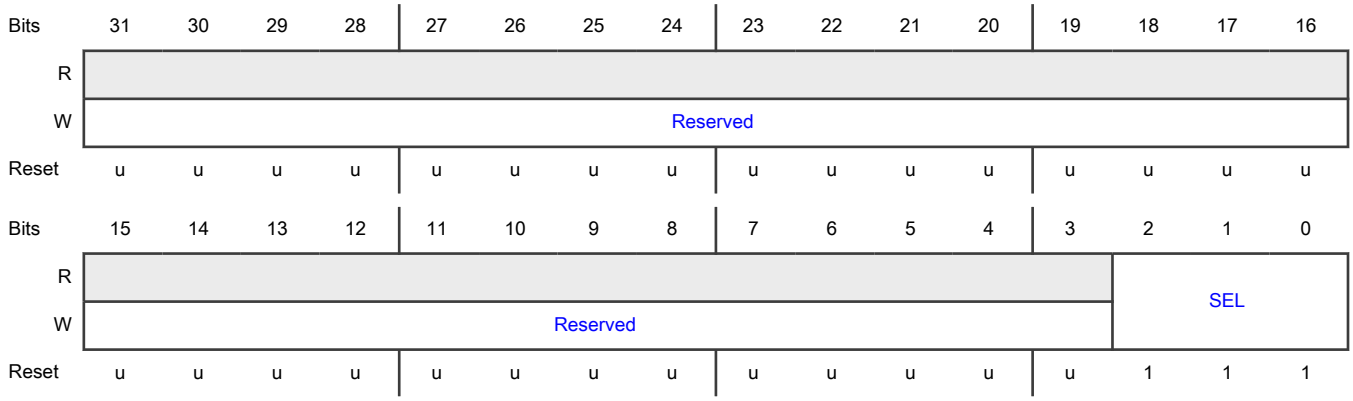
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the System Tick Timer for CPU0 source 000b - SYSTICKCLKDIV0 output 001b - Clk 1 MHz clock 010b - LP Oscillator clock 011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.20 CPU1 System Tick Timer Source Select (SYSTICKCLKSEL1)

Offset

Register	Offset
SYSTICKCLKSEL1	264h

Diagram



Fields

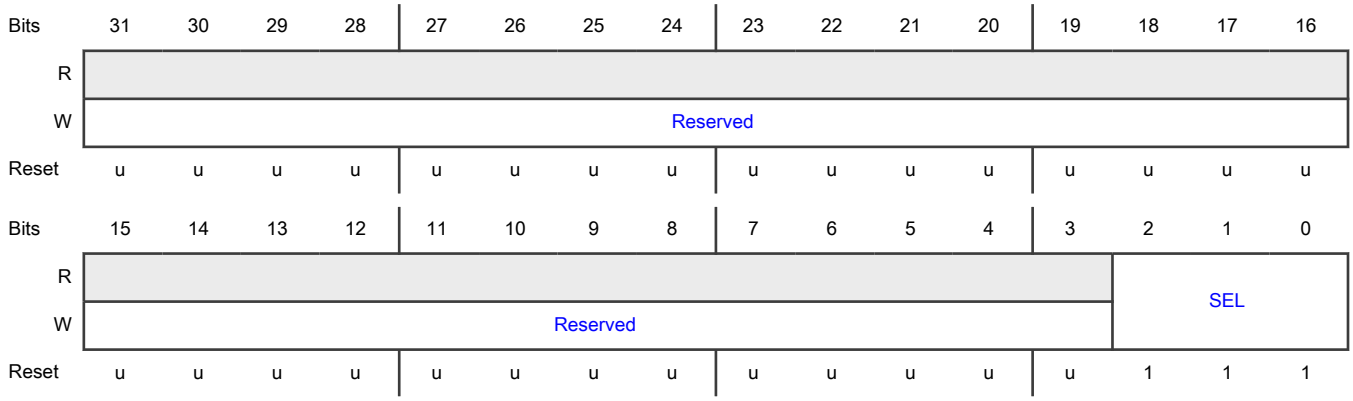
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the System Tick Timer for CPU1 source. 000b - SYSTICKCLKDIV1 output 001b - Clk 1 MHz clock 010b - LP Oscillator clock 011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.21 Trace Clock Source Select (TRACECLKSEL)

Offset

Register	Offset
TRACECLKSEL	268h

Diagram



Fields

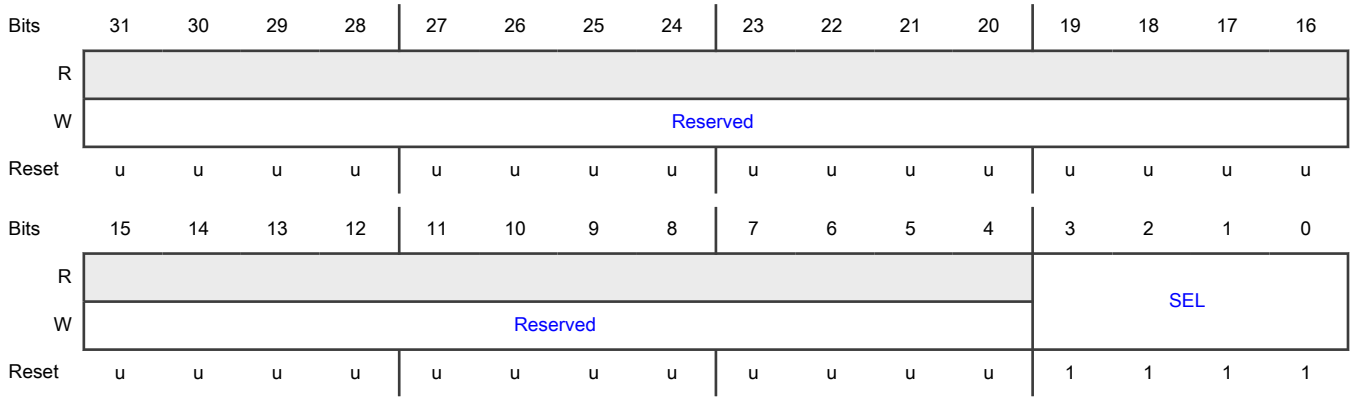
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the trace clock source. 000b - TRACECLKDIV output 001b - Clk 1 MHz clock 010b - LP Oscillator clock 011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.22 CTIMER Clock Source Select (CTIMERCLKSEL0 - CTIMERCLKSEL4)

Offset

Register	Offset
CTIMERCLKSEL0	26Ch
CTIMERCLKSEL1	270h
CTIMERCLKSEL2	274h
CTIMERCLKSEL3	278h
CTIMERCLKSEL4	27Ch

Diagram



Fields

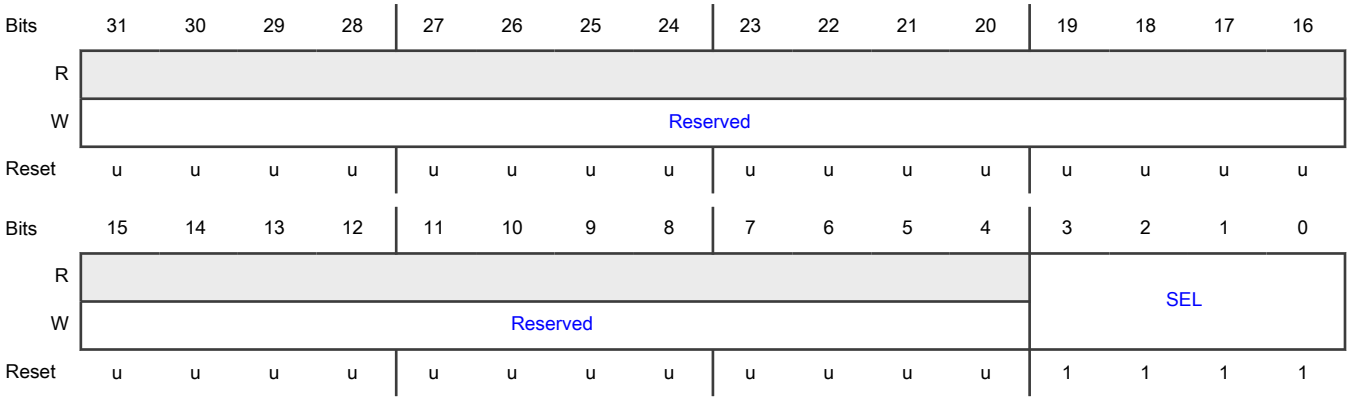
Field	Function
31-4 —	Reserved
3-0 SEL	Selects the CTIMER clock source. 0000b - FRO_1M clock 0001b - PLL0 clock 0010b - PLL1_clk0 clock 0011b - FRO_HF clock 0100b - FRO 12MHz clock 0101b - SAI0 MCLK IN clock 0110b - LP Oscillator clock 0111b - No clock 1000b - SAI1 MCLK IN clock 1001b - SAI0 TX_BCLK clock 1010b - SAI0 RX_BCLK clock 1011b - SAI1 TX_BCLK clock 1100b - SAI1 RX_BCLK clock 1101b - No clock 1110b - No clock 1111b - No clock

17.4.1.23 CLKOUT Clock Source Select (CLKOUTSEL)

Offset

Register	Offset
CLKOUTSEL	288h

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 SEL	Selects the CLKOUT clock source. 0000b - Main clock (main_clk) 0001b - PLL0 clock (pll0_clk) 0010b - CLKIN clock (clk_in) 0011b - FRO_HF clock (fro_hf) 0100b - FRO 12 MHz clock (fro_12m) 0101b - PLL1_clk0 clock (pll1_clk) 0110b - LP Oscillator clock (lp_osc) 0111b - USB PLL clock (usb_pll_clk) 1000b - No clock 1001b - No clock 1010b - No clock 1011b - No clock 1100b - No clock

Table continues on the next page...

Table continued from the previous page...

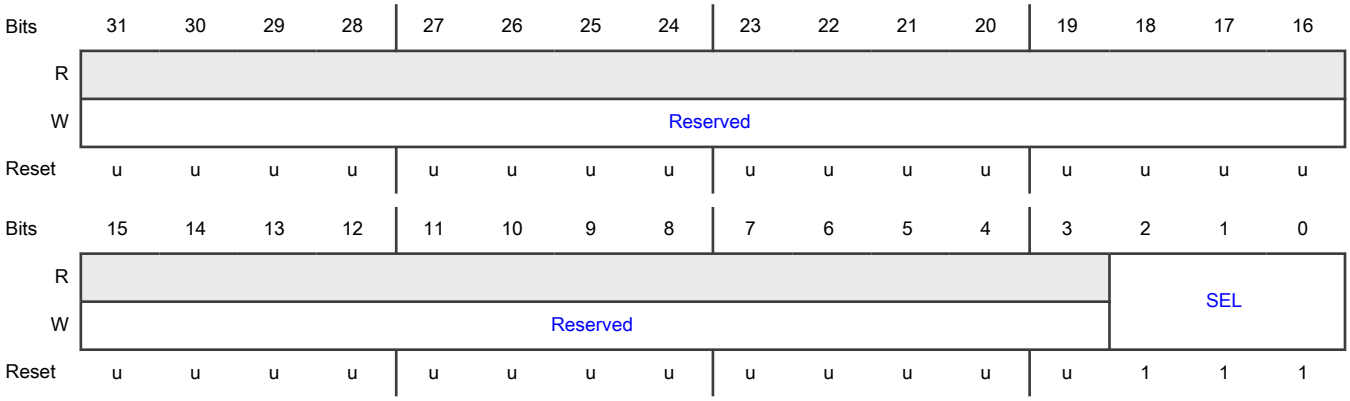
Field	Function
	1101b - No clock 1110b - No clock 1111b - No clock

17.4.1.24 ADC0 Clock Source Select (ADC0CLKSEL)

Offset

Register	Offset
ADC0CLKSEL	2A4h

Diagram



Fields

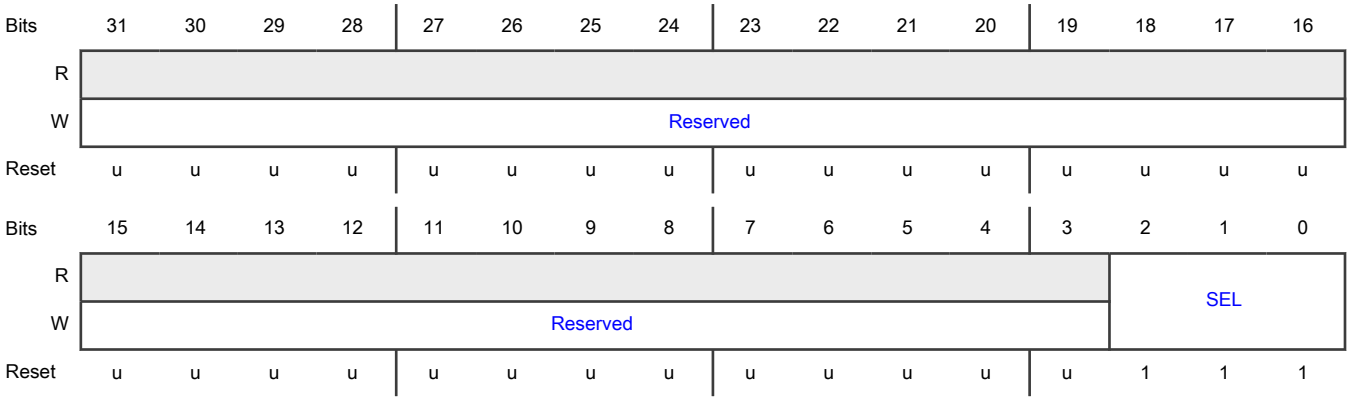
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the ADC0 clock source. 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO 12 MHz clock 100b - Clk_in 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.25 USB-FS Clock Source Select (USB0CLKSEL)

Offset

Register	Offset
USB0CLKSEL	2A8h

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Selects the USB-FS clock source. 000b - No clock 001b - PLL0 clock 010b - No clock 011b - Clk 48 MHz clock 100b - Clk_in 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

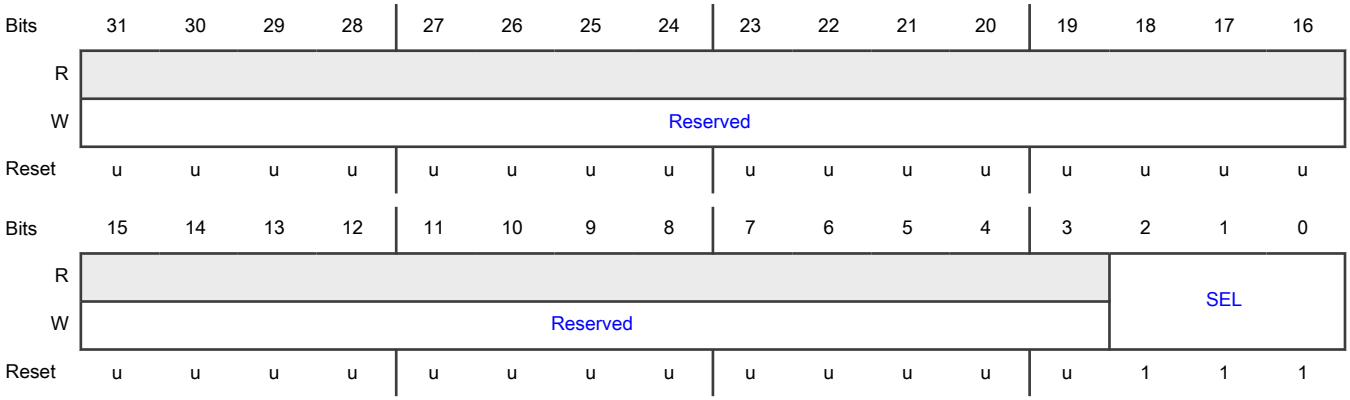
17.4.1.26 LP_FLEXCOMM Clock Source Select for Fractional Rate Divider (FCCLKSEL0 - FCCLKSEL9)

Offset

For n = 0 to 9:

Register	Offset
FCCLKSELn	2B0h + (n × 4h)

Diagram



Fields

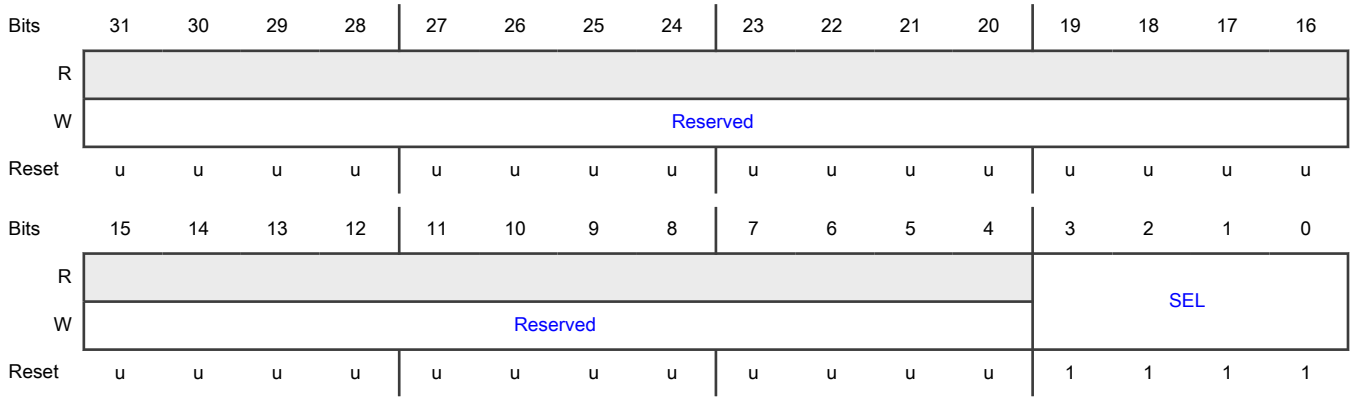
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the LP_FLEXCOMM clock source for Fractional Rate Divider. 000b - No clock 001b - PLL divided clock 010b - FRO 12 MHz clock 011b - fro_hf_div clock 100b - clk_1m clock 101b - USB PLL clock 110b - LP Oscillator clock 111b - No clock

17.4.1.27 SCTimer/PWM Clock Source Select (SCTCLKSEL)

Offset

Register	Offset
SCTCLKSEL	2F0h

Diagram



Fields

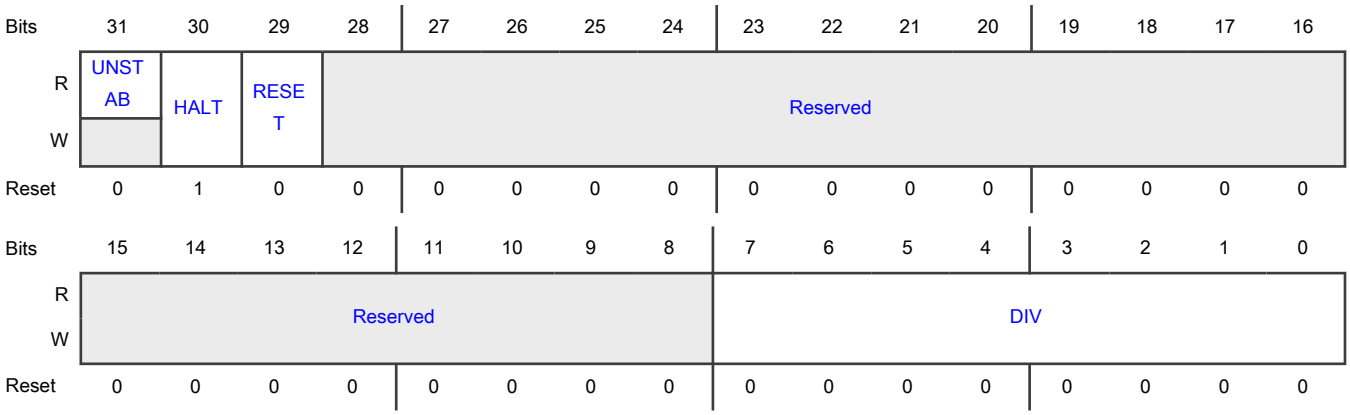
Field	Function
31-4 —	Reserved
3-0 SEL	Selects the SCTimer/PWM clock source. 0000b - No clock 0001b - PLL0 clock 0010b - CLKIN clock 0011b - FRO_HF clock 0100b - PLL1_clk0 clock 0101b - SAI0 MCLK_IN clock 0110b - USB PLL clock 0111b - No clock 1000b - SAI1 MCLK_IN clock 1001b - No clock 1010b - No clock 1011b - No clock 1100b - No clock 1101b - No clock 1110b - No clock 1111b - No clock

17.4.1.28 CPU0 System Tick Timer Divider (SYSTICKCLKDIV0)

Offset

Register	Offset
SYSTICKCLKDIV0	300h

Diagram



Fields

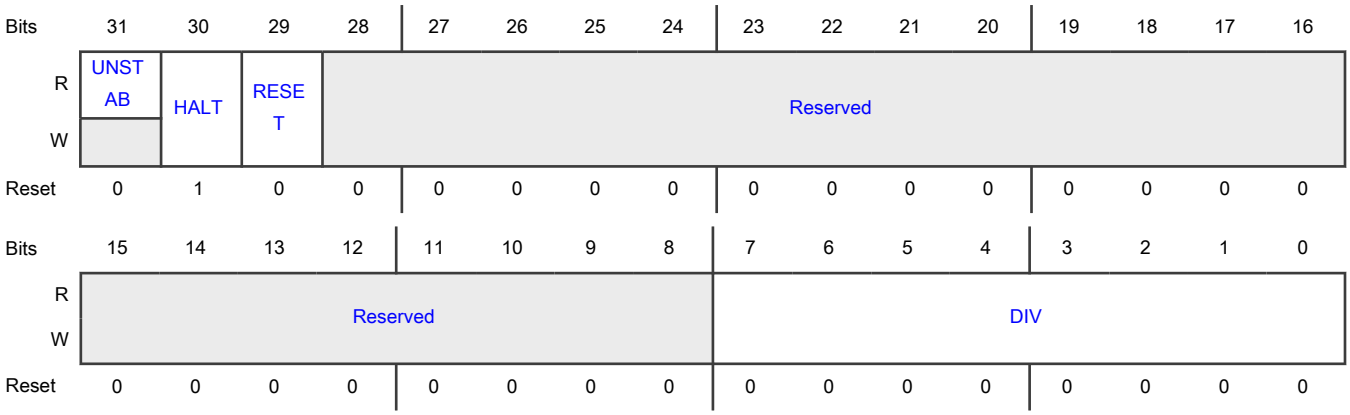
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset.
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.29 CPU1 System Tick Timer Divider (SYSTICKCLKDIV1)

Offset

Register	Offset
SYSTICKCLKDIV1	304h

Diagram



Fields

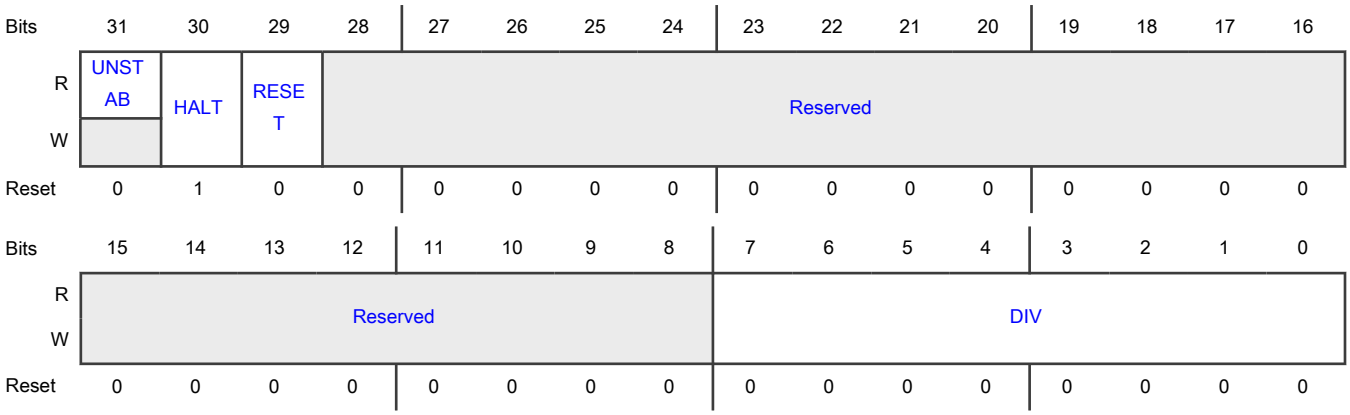
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.30 TRACE Clock Divider (TRACECLKDIV)

Offset

Register	Offset
TRACECLKDIV	308h

Diagram



Fields

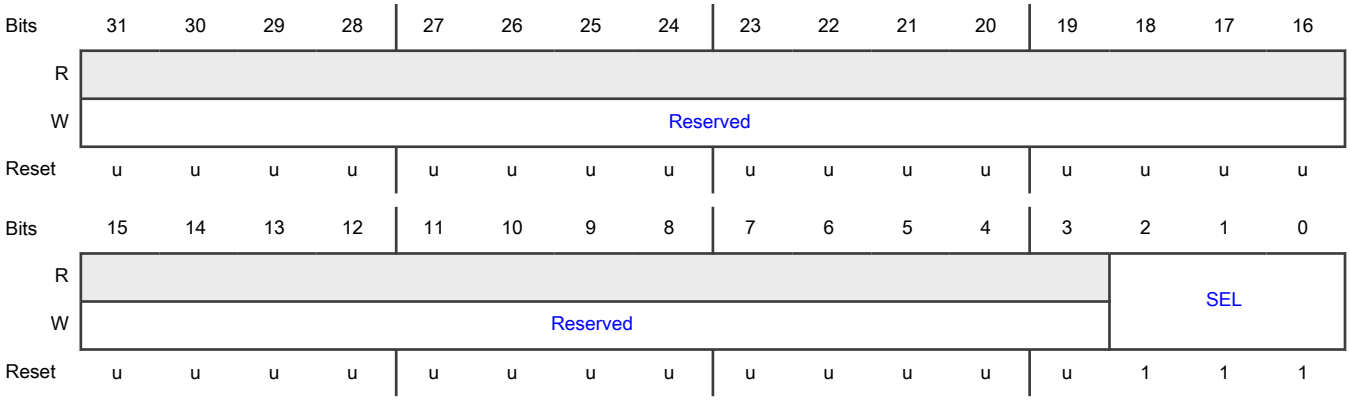
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.31 TSI Function Clock Source Select (TSICLKSEL)

Offset

Register	Offset
TSICLKSEL	350h

Diagram



Fields

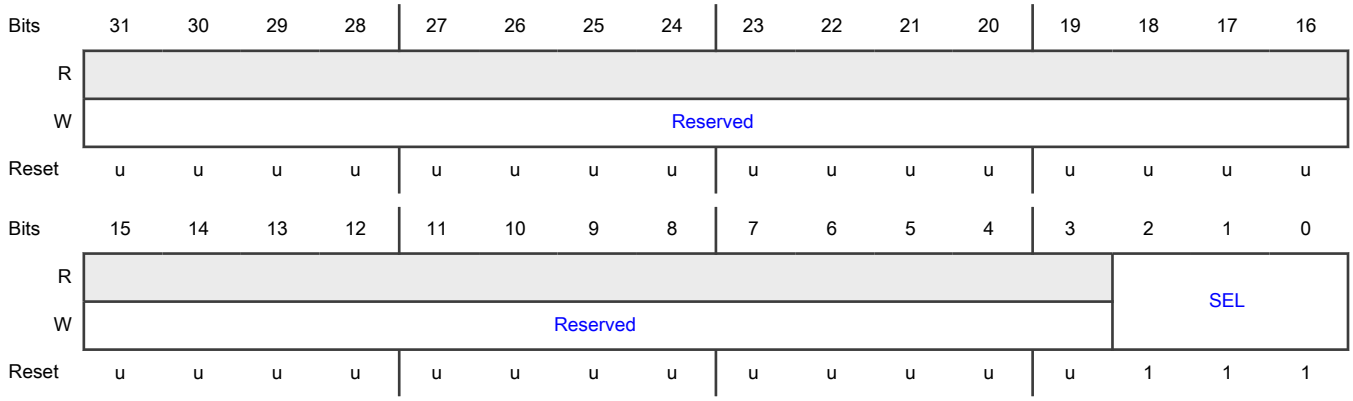
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the TSI function clock source. 000b - No clock 001b - No clock 010b - clk_in 011b - No clock 100b - FRO_12Mhz clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.32 SINC FILTER Function Clock Source Select (SINCFILTCLKSEL)

Offset

Register	Offset
SINCFILTCLKSEL	360h

Diagram



Fields

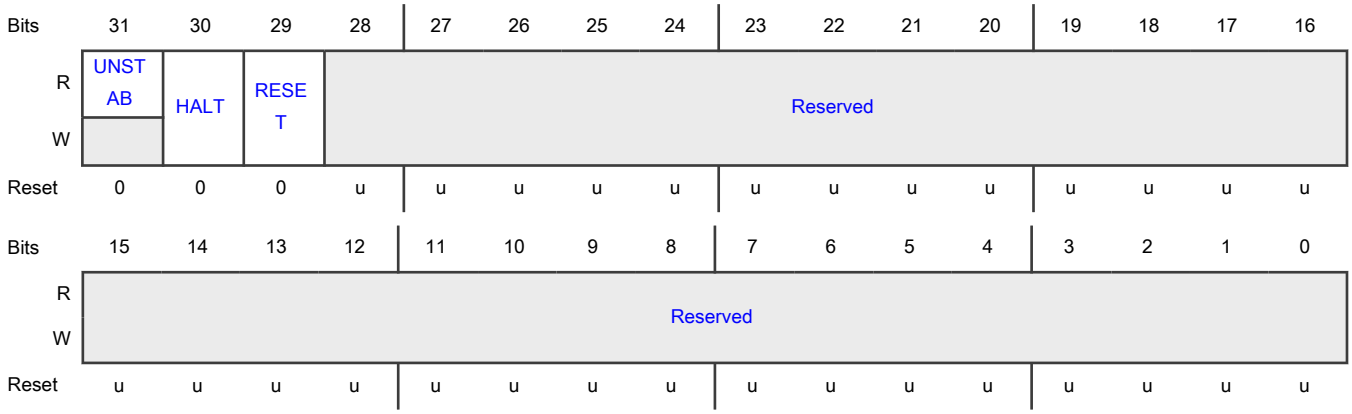
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the SINC FILTER function clock source. 000b - No clock 001b - PLL0 clock 010b - clk_in 011b - FRO_HF clock 100b - FRO_12Mhz clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.33 SLOW_CLK Clock Divider (SLOWCLKDIV)

Offset

Register	Offset
SLOWCLKDIV	378h

Diagram



Fields

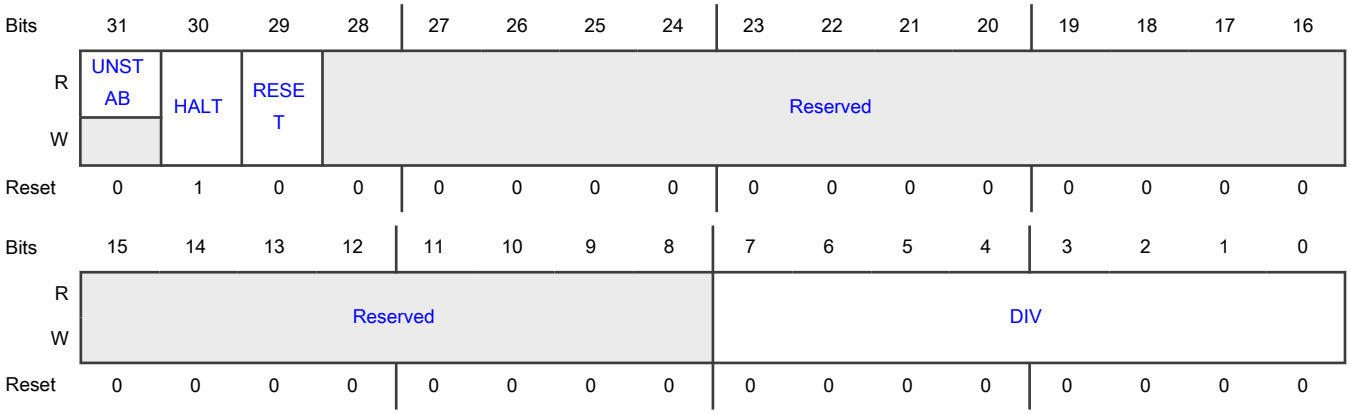
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-0 —	Reserved

17.4.1.34 TSI Function Clock Divider (TSICLKDIV)

Offset

Register	Offset
TSICLKDIV	37Ch

Diagram



Fields

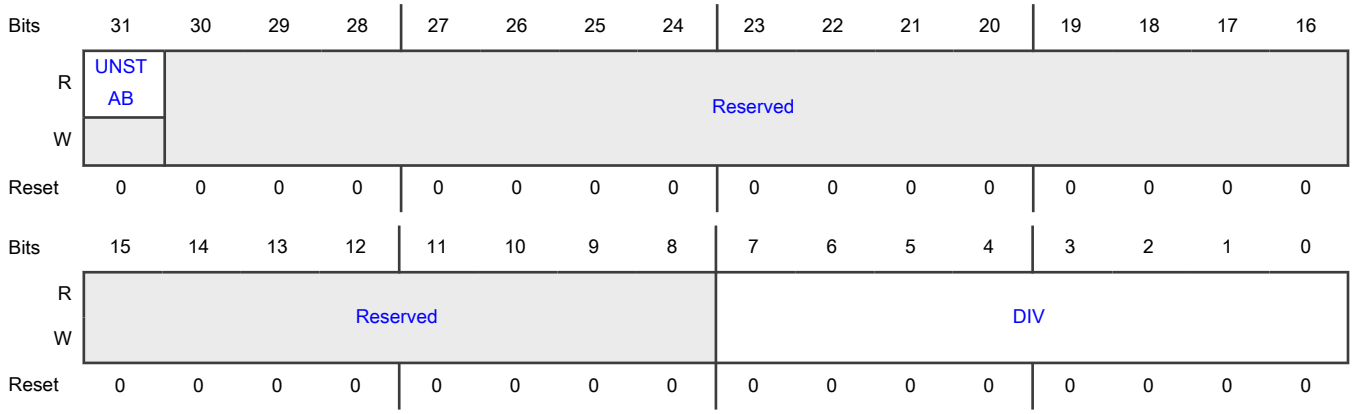
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value: • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.35 System Clock Divider (AHBCLKDIV)

Offset

Register	Offset
AHBCLKDIV	380h

Diagram



Fields

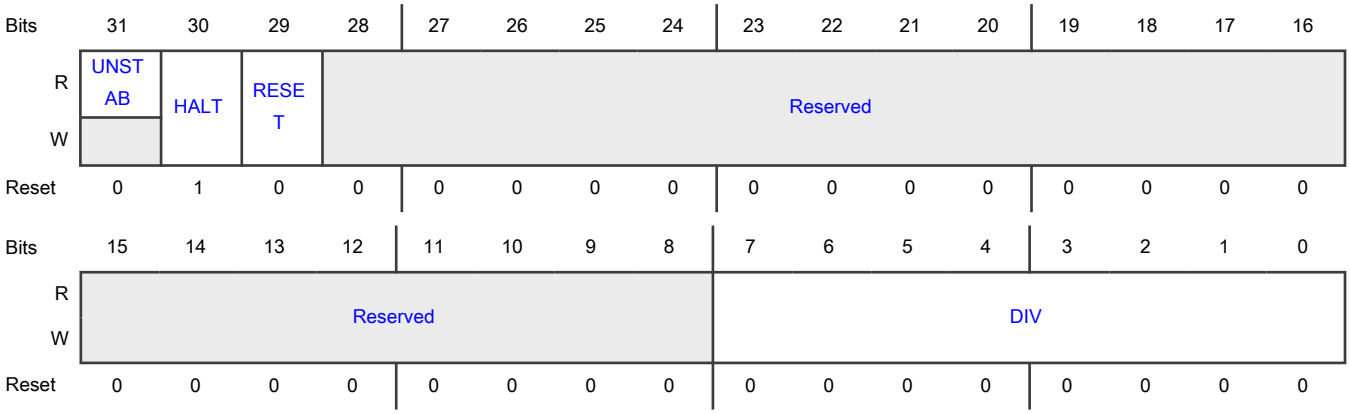
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.36 CLKOUT Clock Divider (CLKOUTDIV)

Offset

Register	Offset
CLKOUTDIV	384h

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.37 FRO_HF_DIV Clock Divider (FROHFDIV)

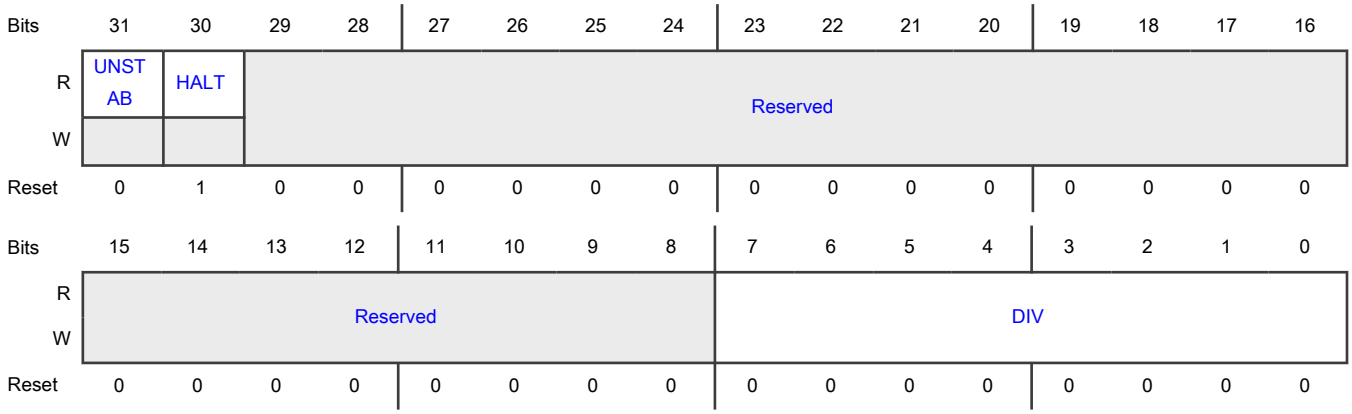
Offset

Register	Offset
FROHFDIV	388h

Function

This register is used to generate fro_hf_div clock from fro_hf clock.

Diagram



Fields

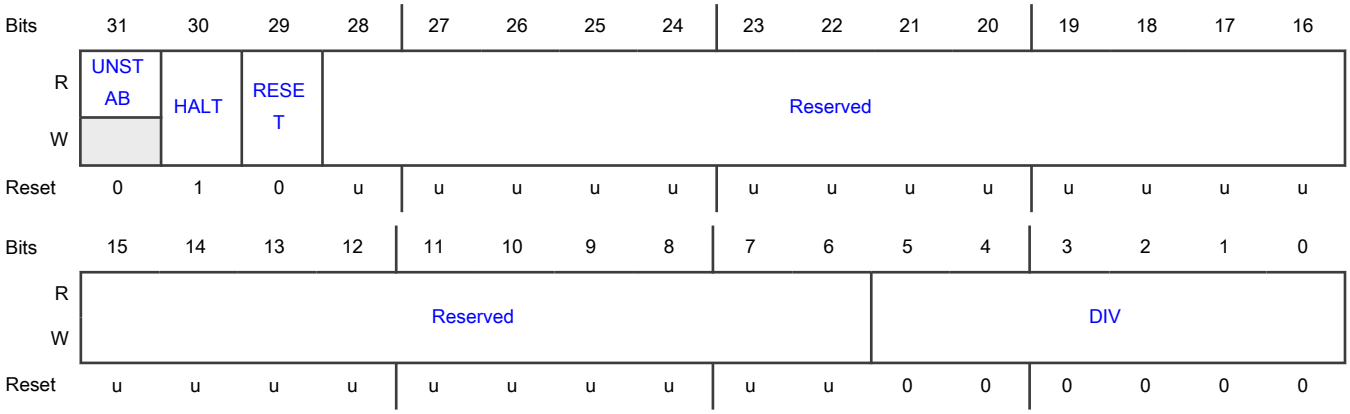
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running, this bit is set to 0 when the register is written. 1b - Divider clock is stopped
29-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.38 WDT0 Clock Divider (WDT0CLKDIV)

Offset

Register	Offset
WDT0CLKDIV	38Ch

Diagram



Fields

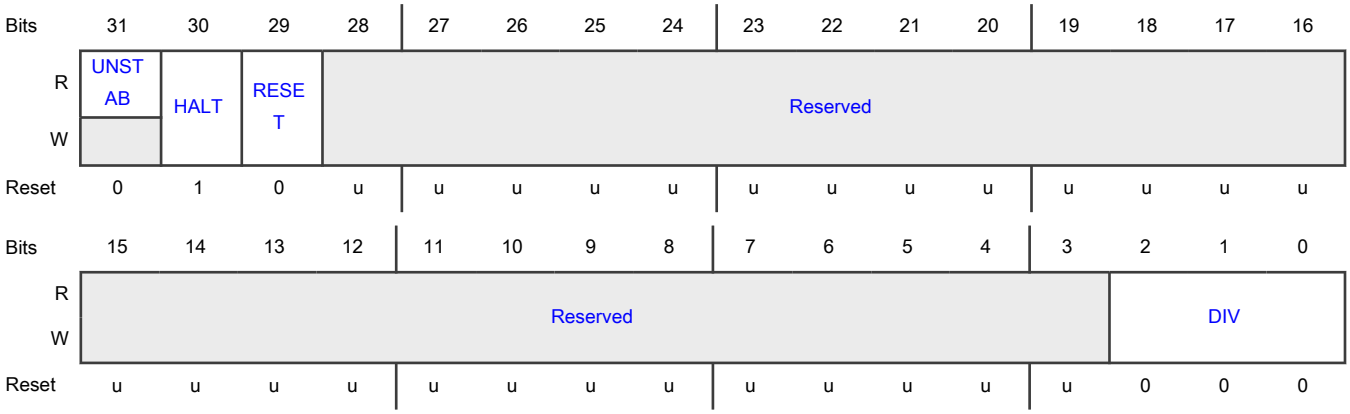
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-6 —	Reserved
5-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.39 ADC0 Clock Divider (ADC0CLKDIV)

Offset

Register	Offset
ADC0CLKDIV	394h

Diagram



Fields

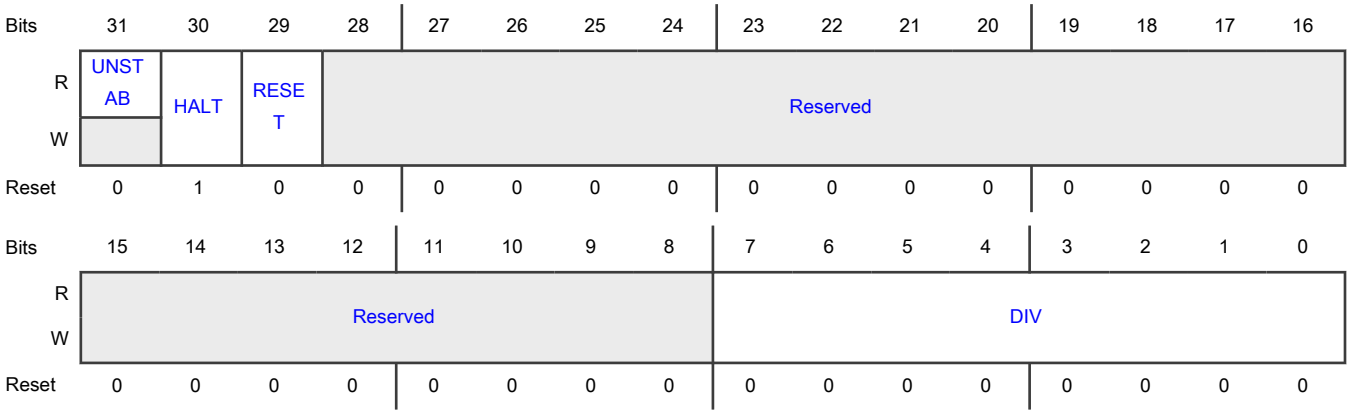
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.40 USB-FS Clock Divider (USB0CLKDIV)

Offset

Register	Offset
USB0CLKDIV	398h

Diagram



Fields

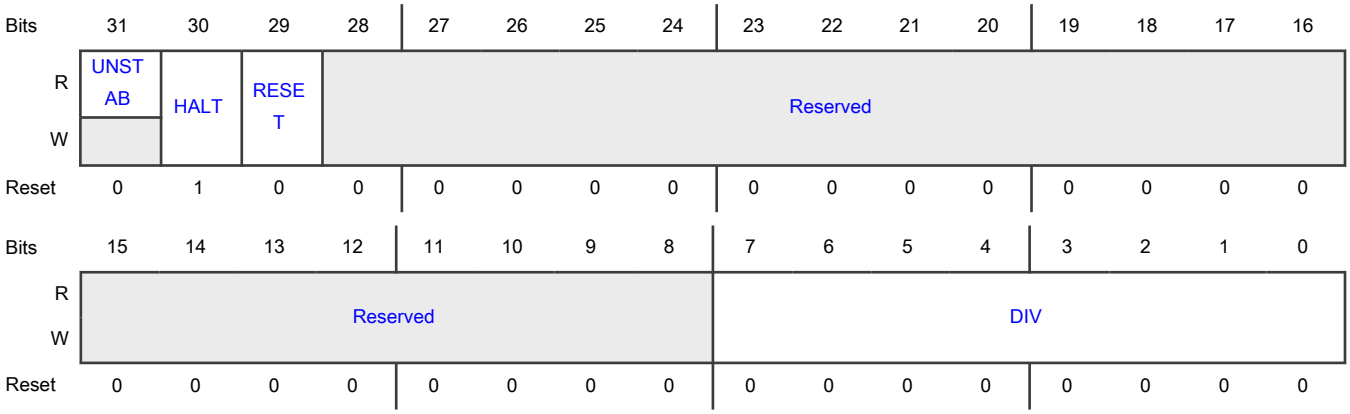
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.41 SCT/PWM Clock Divider (SCTCLKDIV)

Offset

Register	Offset
SCTCLKDIV	3B4h

Diagram



Fields

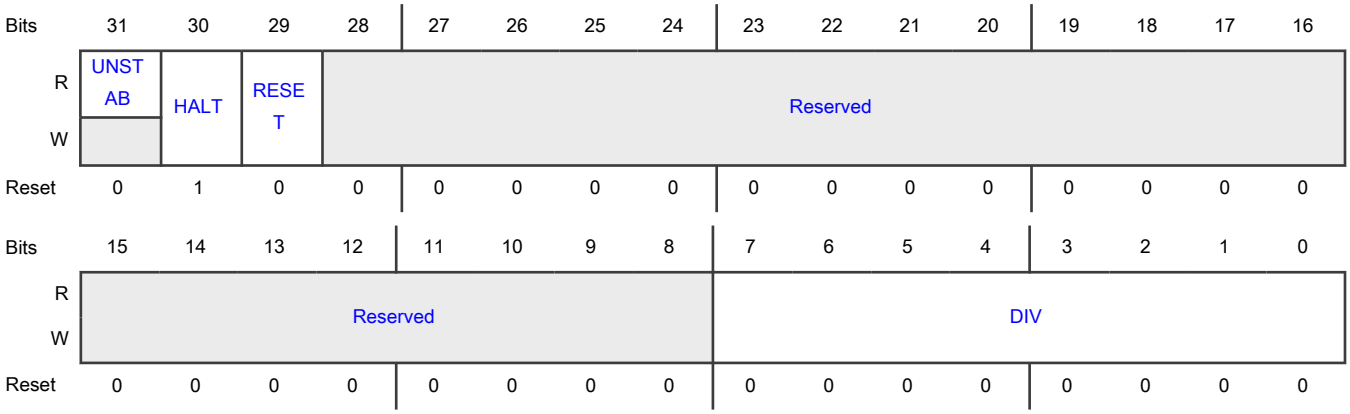
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.42 PLL Clock Divider (PLLCLKDIV)

Offset

Register	Offset
PLLCLKDIV	3C4h

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.43 CTimer Clock Divider (CTIMER0CLKDIV - CTIMER4CLKDIV)

Offset

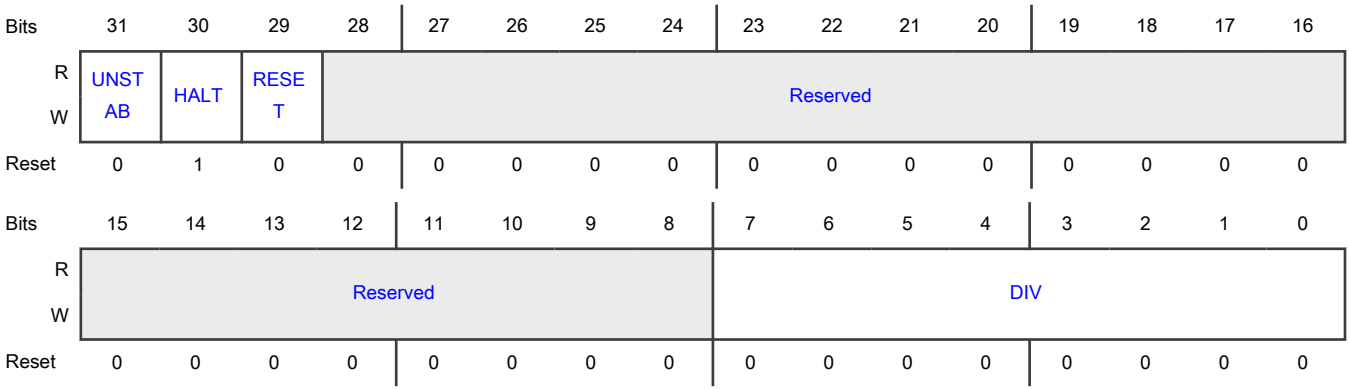
Register	Offset
CTIMER0CLKDIV	3D0h
CTIMER1CLKDIV	3D4h
CTIMER2CLKDIV	3D8h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
CTIMER3CLKDIV	3DCh
CTIMER4CLKDIV	3E0h

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Stable divider clock 1b - Unstable clock frequency
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock has stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.44 PLL1 Clock 0 Divider (PLL1CLK0DIV)

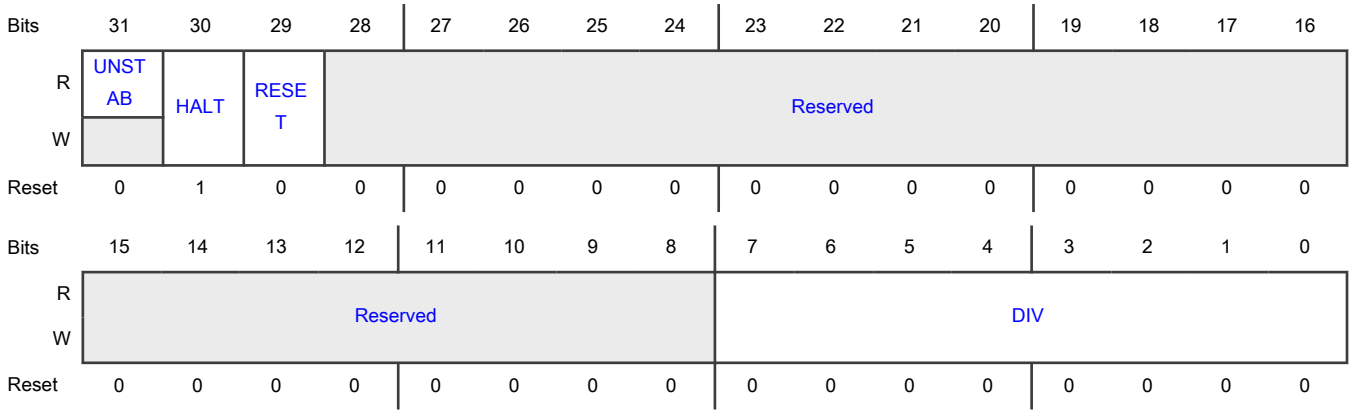
Offset

Register	Offset
PLL1CLK0DIV	3E4h

Function

This register is used to generate pll1_clk0 clock from pll1_clk clock.

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.45 PLL1 Clock 1 Divider (PLL1CLK1DIV)

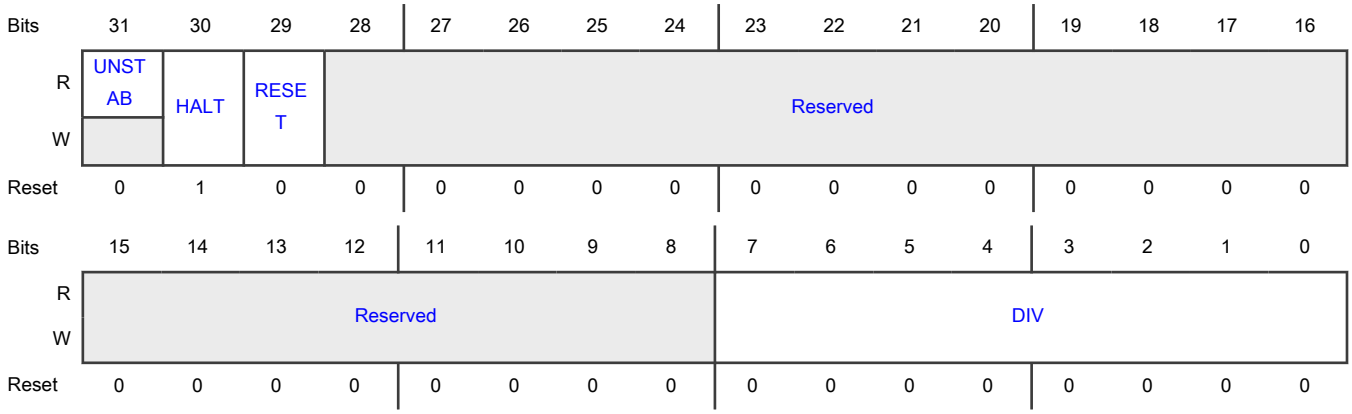
Offset

Register	Offset
PLL1CLK1DIV	3E8h

Function

This register is used to generate pll1_clk1 clock from pll1_clk clock.

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

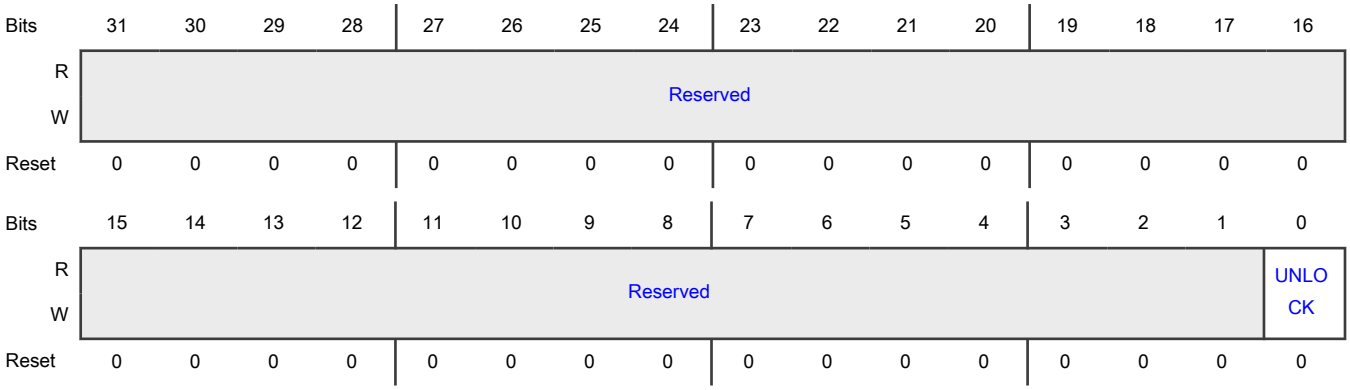
17.4.1.46 Clock Configuration Unlock (CLKUNLOCK)

Offset

Register	Offset
CLKUNLOCK	3FCh

Function
This register controls access to the clock select and divider configuration registers.

Diagram



Fields

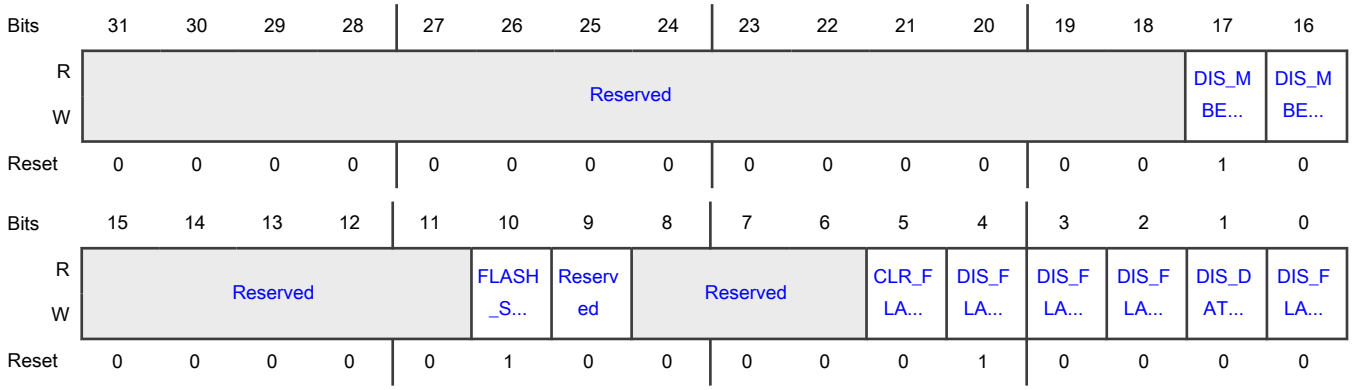
Field	Function
31-1 —	Reserved
0 UNLOCK	Controls clock configuration registers access (for example, xxxDIV, xxxSEL) 0b - Updates are allowed to all clock configuration registers 1b - Freezes all clock configuration registers update

17.4.1.47 NVM Control (NVM_CTRL)

Offset

Register	Offset
NVM_CTRL	400h

Diagram



Fields

Field	Function
31-18 —	Reserved
17 DIS_MBECC_ERR_DATA	Bus error on data multi-bit ECC error control 0b - Enables bus error on multi-bit ECC error for data 1b - Disables bus error on multi-bit ECC error for data
16 DIS_MBECC_ERR_INST	Bus error on instruction multi-bit ECC error control 0b - Enables bus error on multi-bit ECC error for instruction 1b - Disables bus error on multi-bit ECC error for instruction
15-11 —	Reserved
10 FLASH_STALL_EN	FLASH stall on busy control 0b - No stall on FLASH busy 1b - Stall on FLASH busy
9 —	Reserved Keep the default value.
8-6 —	Reserved
5 CLR_FLASH_CACHE	Clear flash cache control 0b - No clear flash cache 1b - Clears flash cache
4	Flash data cache control

Table continues on the next page...

Table continued from the previous page...

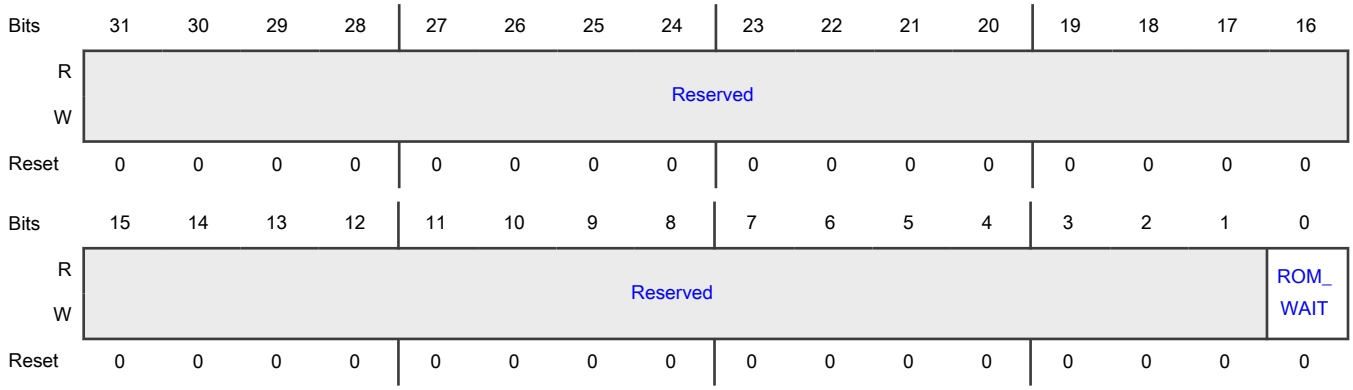
Field	Function
DIS_FLASH_DATA	0b - Enables flash data cache when DIS_FLASH_CACHE=0 1b - Disables flash data cache
3 DIS_FLASH_INST	Flash instruction cache control 0b - Enables flash instruction cache when DIS_FLASH_CACHE=0 1b - Disables flash instruction cache
2 DIS_FLASH_CACHE	Flash cache control 0b - Enables flash cache 1b - Disables flash cache
1 DIS_DATA_SPEC	Flash data speculation control **NOTE** If DIS_MBECC_ERR_DATA and/or DIS_MBECC_ERR_INST are set, then speculation will not be enabled, even if this bit is cleared. 0b - Enables data speculation 1b - Disables data speculation
0 DIS_FLASH_SPEC	Flash speculation control **NOTE** If DIS_MBECC_ERR_DATA and/or DIS_MBECC_ERR_INST are set, then speculation will not be enabled, even if this bit is cleared. 0b - Enables flash speculation 1b - Disables flash speculation

17.4.1.48 ROM Wait State (ROMCR)

Offset

Register	Offset
ROMCR	404h

Diagram



Fields

Field	Function
31-1 —	Reserved
0 ROM_WAIT	ROM waiting Arm core and other masters for one cycle 0b - Disabled 1b - Enabled

17.4.1.49 SmartDMA Interrupt Hijack (SmartDMAINT)

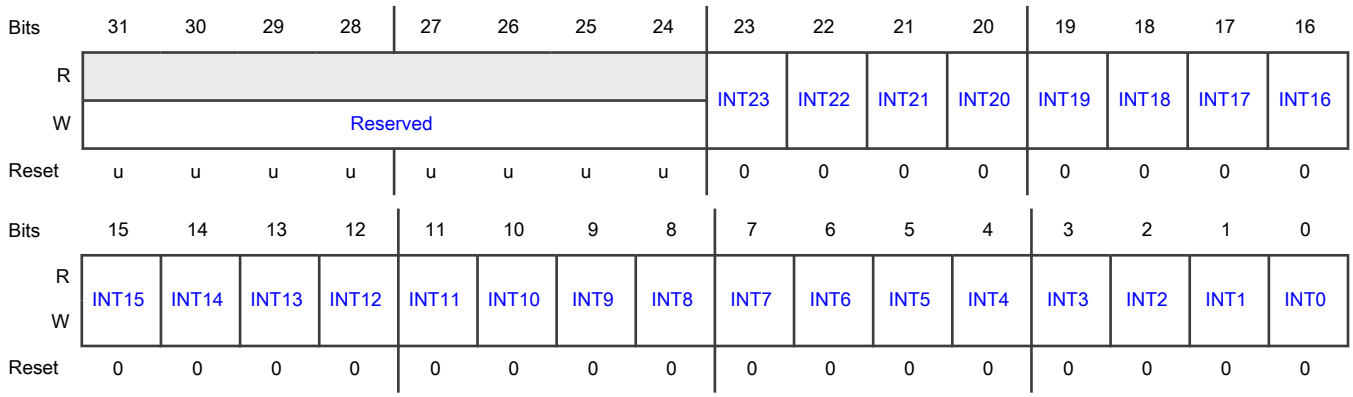
Offset

Register	Offset
SmartDMAINT	414h

Function

Bits directly control the SmartDMA hijacking the system interrupts.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 INT23	SmartDMA hijack NVIC IRQ77 0b - Disable 1b - Enable
22 INT22	SmartDMA hijack NVIC IRQ67 0b - Disable 1b - Enable
21 INT21	SmartDMA hijack NVIC IRQ66 0b - Disable 1b - Enable
20 INT20	SmartDMA hijack NVIC IRQ51 0b - Disable 1b - Enable
19 INT19	SmartDMA hijack NVIC IRQ50 0b - Disable 1b - Enable
18 INT18	SmartDMA hijack NVIC IRQ47 0b - Disable 1b - Enable
17 INT17	SmartDMA hijack NVIC IRQ45 0b - Disable 1b - Enable
16 INT16	SmartDMA hijack NVIC IRQ42 0b - Disable 1b - Enable
15 INT15	SmartDMA hijack NVIC IRQ41 0b - Disable 1b - Enable
14 INT14	SmartDMA hijack NVIC IRQ40 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 INT13	SmartDMA hijack NVIC IRQ39 0b - Disable 1b - Enable
12 INT12	SmartDMA hijack NVIC IRQ38 0b - Disable 1b - Enable
11 INT11	SmartDMA hijack NVIC IRQ37 0b - Disable 1b - Enable
10 INT10	SmartDMA hijack NVIC IRQ36 0b - Disable 1b - Enable
9 INT9	SmartDMA hijack NVIC IRQ35 0b - Disable 1b - Enable
8 INT8	SmartDMA hijack NVIC IRQ34 0b - Disable 1b - Enable
7 INT7	SmartDMA hijack NVIC IRQ33 0b - Disable 1b - Enable
6 INT6	SmartDMA hijack NVIC IRQ32 0b - Disable 1b - Enable
5 INT5	SmartDMA hijack NVIC IRQ31 0b - Disable 1b - Enable
4 INT4	SmartDMA hijack NVIC IRQ30 0b - Disable 1b - Enable
3 INT3	SmartDMA hijack NVIC IRQ29

Table continues on the next page...

Table continued from the previous page...

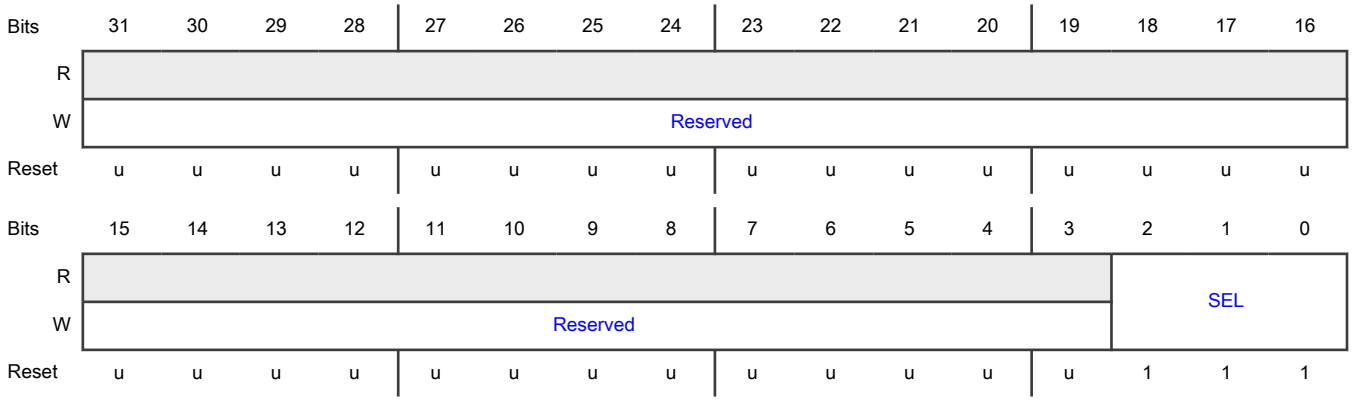
Field	Function
	0b - Disable 1b - Enable
2 INT2	SmartDMA hijack NVIC IRQ18 0b - Disable 1b - Enable
1 INT1	SmartDMA hijack NVIC IRQ17 0b - Disable 1b - Enable
0 INT0	SmartDMA hijack NVIC IRQ1 0b - Disable 1b - Enable

17.4.1.50 ADC1 Clock Source Select (ADC1CLKSEL)

Offset

Register	Offset
ADC1CLKSEL	464h

Diagram



Fields

Field	Function
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

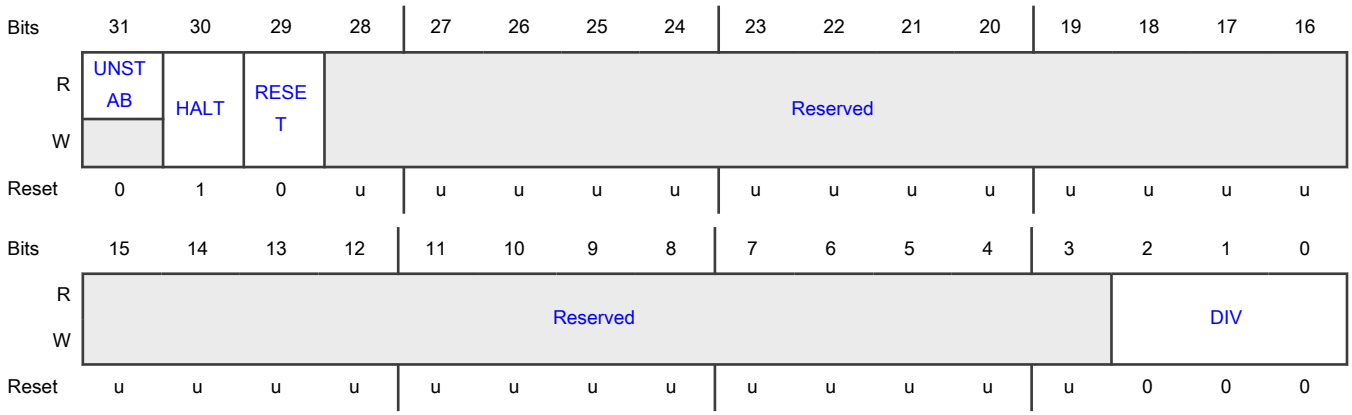
Field	Function
—	
2-0 SEL	Selects the ADC1 clock source 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO 12 MHz clock 100b - Clk_in clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.51 ADC1 Clock Divider (ADC1CLKDIV)

Offset

Register	Offset
ADC1CLKDIV	468h

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable

Table continues on the next page...

Table continued from the previous page...

Field	Function
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.52 DACn Functional Clock Selection (DAC0CLKSEL - DAC2CLKSEL)

Offset

Register	Offset
DAC0CLKSEL	490h
DAC1CLKSEL	498h
DAC2CLKSEL	4A0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved													SEL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

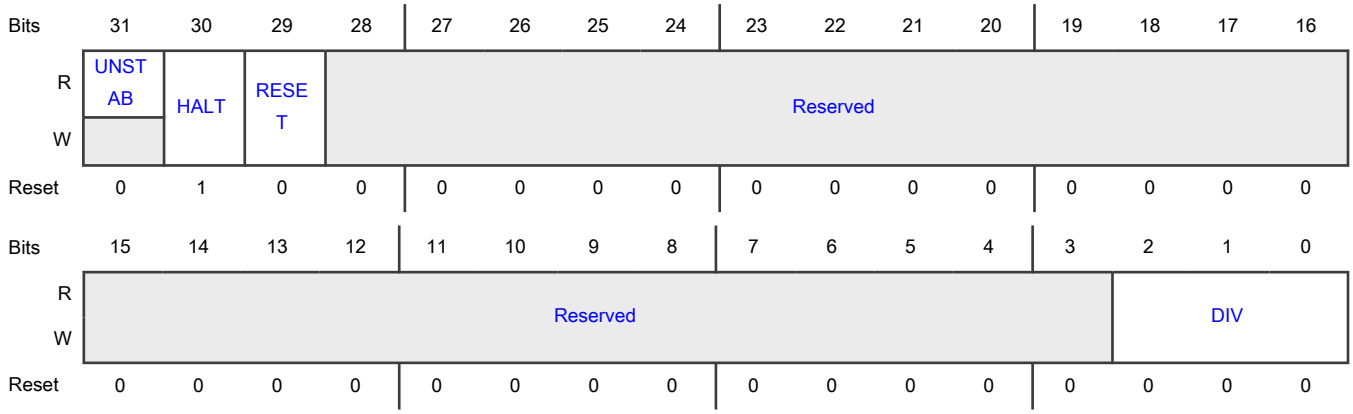
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the DAC clock source 000b - No clock 001b - PLL0 clock 010b - Clk_in 011b - FRO_HF 100b - FRO_12M 101b - PLL1_clk0 clock 110b - No clock 111b - No clock

17.4.1.53 DACn functional clock divider (DAC0CLKDIV - DAC2CLKDIV)

Offset

Register	Offset
DAC0CLKDIV	494h
DAC1CLKDIV	49Ch
DAC2CLKDIV	4A4h

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.54 FlexSPI Clock Selection (FlexSPICLKSEL)

Offset

Register	Offset
FlexSPICLKSEL	4A8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved												SEL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Fields

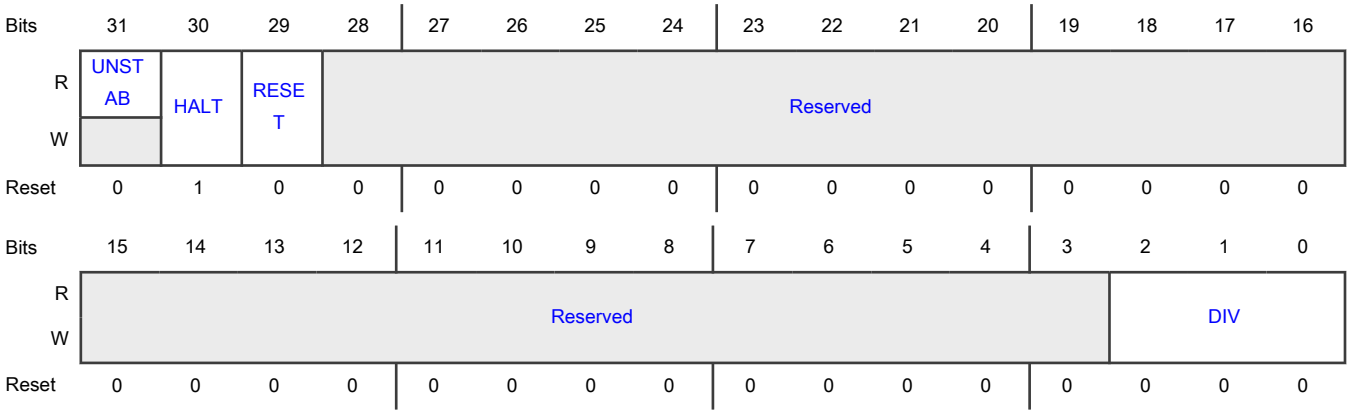
Field	Function
31-4 —	Reserved
3-0 SEL	Selects the FlexSPI clock 0000b - No clock 0001b - PLL0 clock 0010b - No clock 0011b - FRO_HF 0100b - No clock 0101b - pll1_clock 0110b - USB PLL clock 0111b - No clock 1000b - No clock 1001b - No clock 1010b - No clock 1011b - No clock 1100b - No clock 1101b - No clock 1110b - No clock 1111b - No clock

17.4.1.55 FlexSPI Clock Divider (FlexSPICLKDIV)

Offset

Register	Offset
FlexSPICLKDIV	4ACh

Diagram



Fields

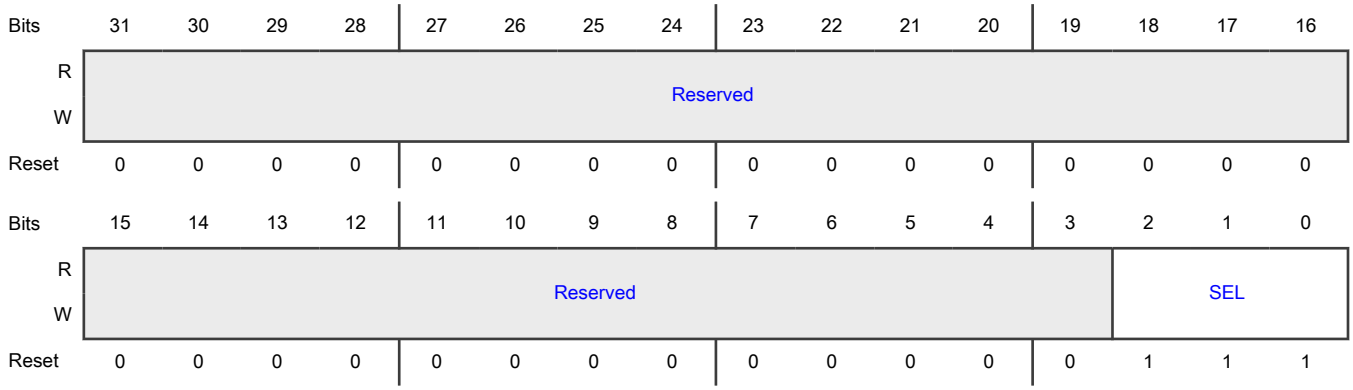
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.56 PLL Clock Divider Clock Selection (PLLCLKDIVSEL)

Offset

Register	Offset
PLLCLKDIVSEL	52Ch

Diagram



Fields

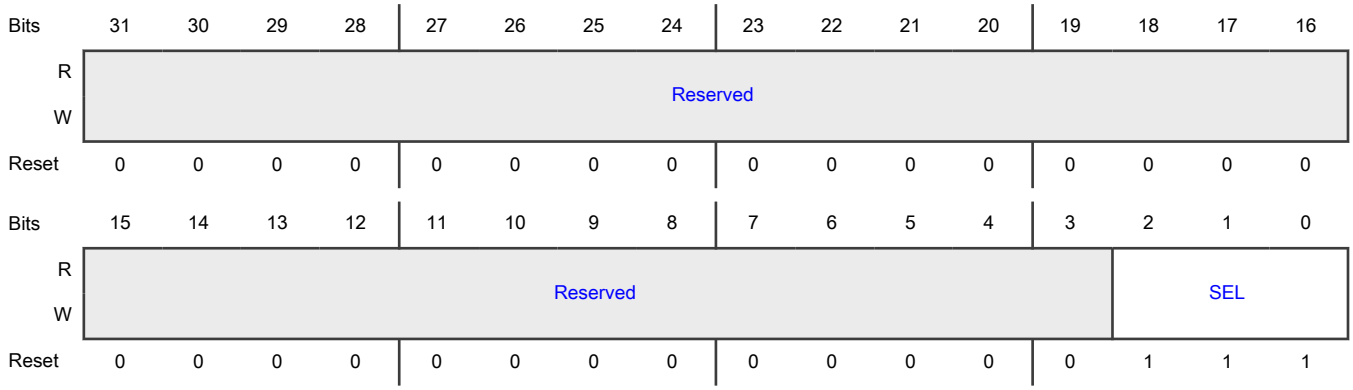
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the PLL Clock Divider source clock 000b - PLL0 clock 001b - pll1_clk0 010b - No clock 011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.57 I3C0 Functional Clock Selection (I3C0FCLKSEL)

Offset

Register	Offset
I3C0FCLKSEL	530h

Diagram



Fields

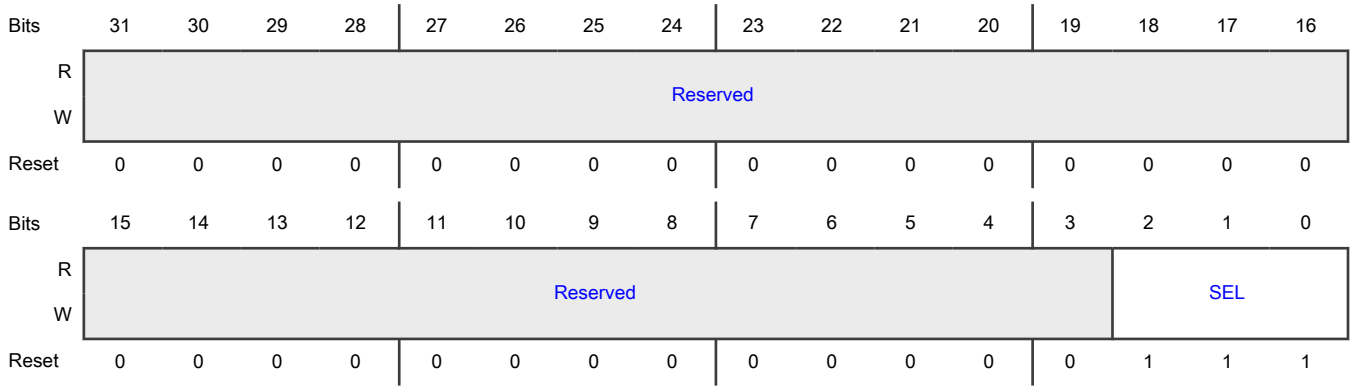
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the I3C0 clock 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - No clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.58 I3C0 FCLK_STC Clock Selection (I3C0FCLKSTCSEL)

Offset

Register	Offset
I3C0FCLKSTCSEL	534h

Diagram



Fields

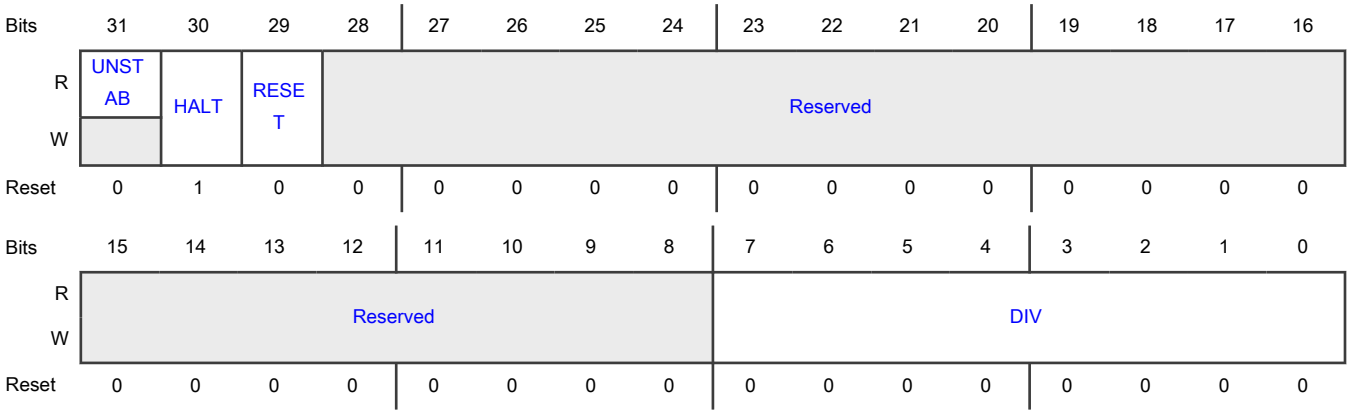
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the I3C0 Time Control clock 000b - I3C0 functional clock I3C0FCLK 001b - FRO_1M clock 010b - No clock 011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.59 I3C0 FCLK_STC Clock Divider (I3C0FCLKSTCDIV)

Offset

Register	Offset
I3C0FCLKSTCDIV	538h

Diagram



Fields

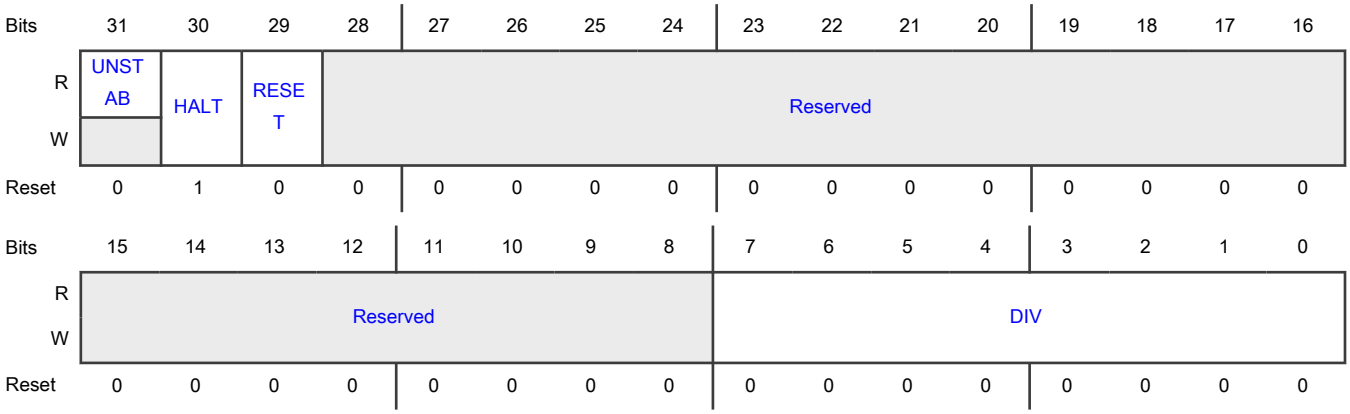
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.60 I3C0 FCLK Slow Clock Divider (I3C0FCLKSDIV)

Offset

Register	Offset
I3C0FCLKSDIV	53Ch

Diagram



Fields

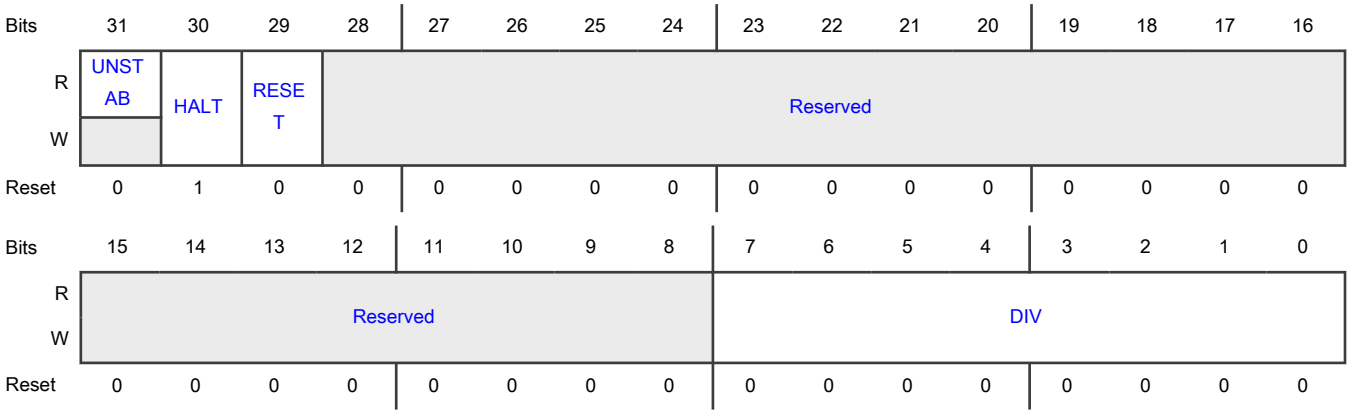
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.61 I3C0 Functional Clock FCLK Divider (I3C0FCLKDIV)

Offset

Register	Offset
I3C0FCLKDIV	540h

Diagram



Fields

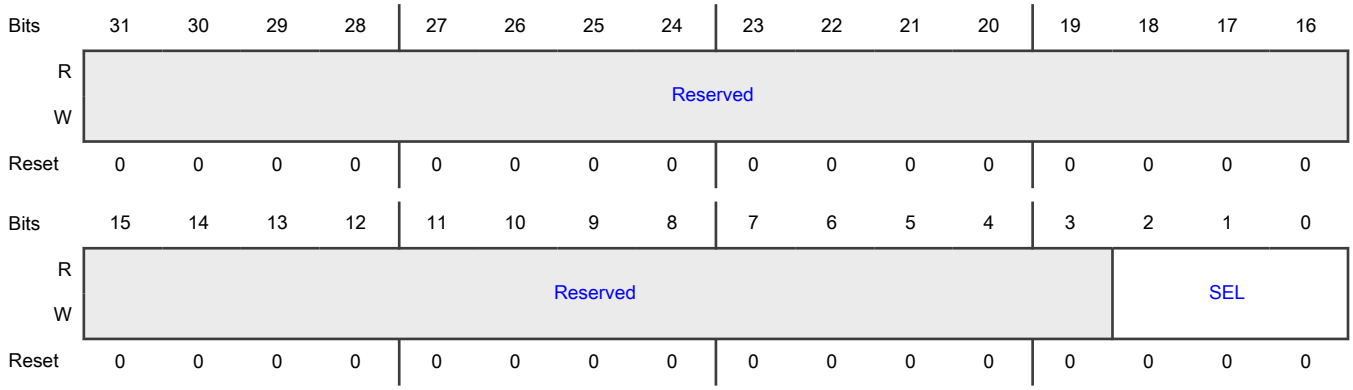
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.62 I3C0 FCLK Slow Selection (I3C0FCLKSSEL)

Offset

Register	Offset
I3C0FCLKSSEL	544h

Diagram



Fields

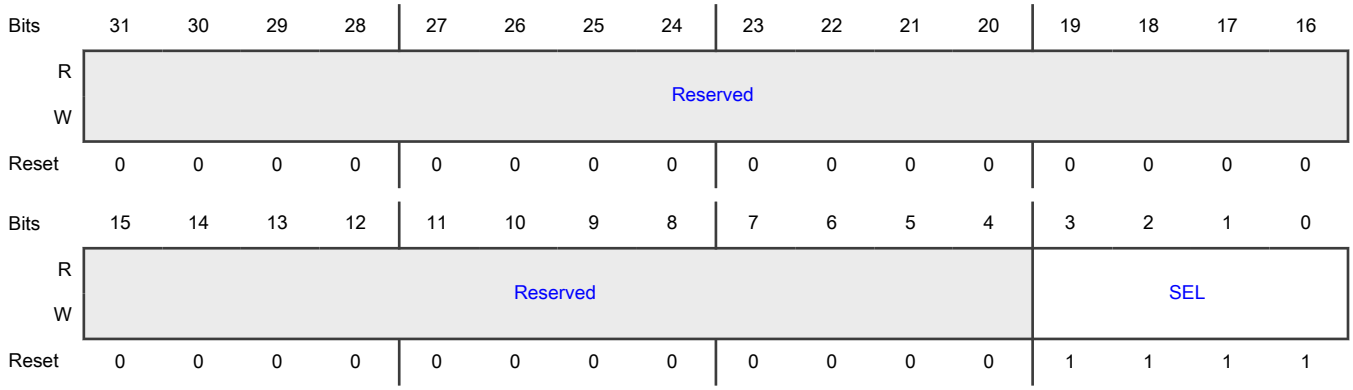
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the I3C FCLK Slow clock 000b - FRO_1M clock 001b - No clock 010b - No clock 011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.63 MICFIL Clock Selection (MICFILFCLKSEL)

Offset

Register	Offset
MICFILFCLKSEL	548h

Diagram



Fields

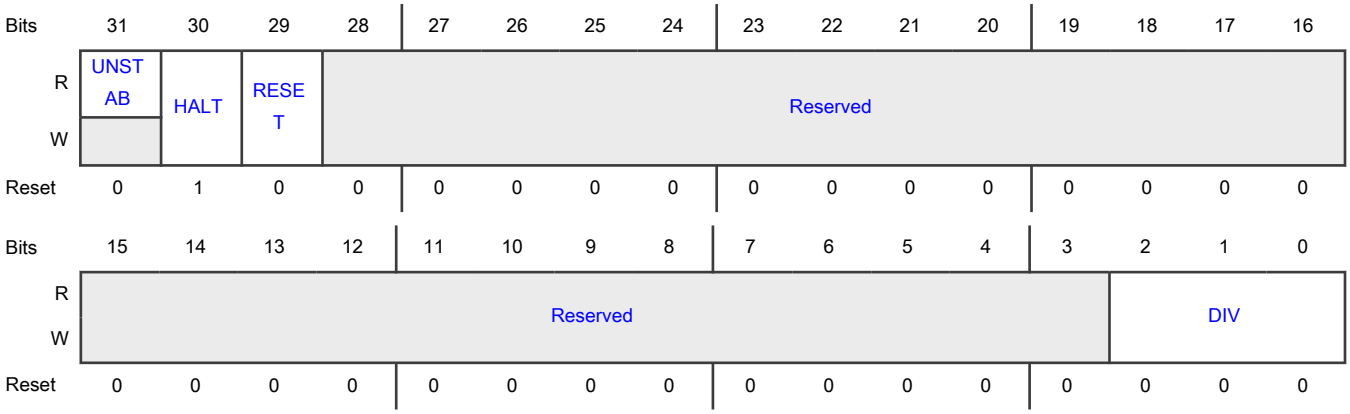
Field	Function
31-4 —	Reserved
3-0 SEL	Selects the MICFIL clock 0000b - FRO_12M clock 0001b - PLL0 clock 0010b - CLKIN clock 0011b - FRO_HF clock 0100b - PLL1_clk0 clock 0101b - SAI0_MCLK clock 0110b - USB PLL clock 0111b - No clock 1000b - SAI1_MCLK clock 1001b - No clock 1010b - No clock 1011b - No clock 1100b - No clock 1101b - No clock 1110b - No clock 1111b - No clock

17.4.1.64 MICFIL Clock Division (MICFILFCLKDIV)

Offset

Register	Offset
MICFILFCLKDIV	54Ch

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.65 uSDHC Clock Selection (uSDHCCLKSEL)

Offset

Register	Offset
uSDHCCLKSEL	558h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved													SEL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

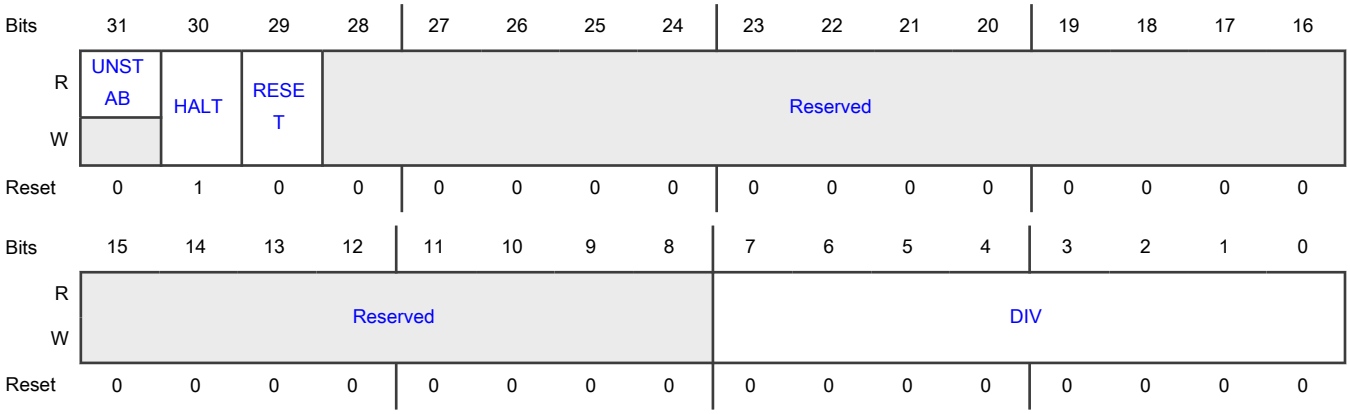
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the uSDHC clock 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - FRO_12M clock 101b - pll1_clk1 clock 110b - USB PLL clock 111b - No clock

17.4.1.66 uSDHC Function Clock Divider (uSDHCCLKDIV)

Offset

Register	Offset
uSDHCCLKDIV	55Ch

Diagram



Fields

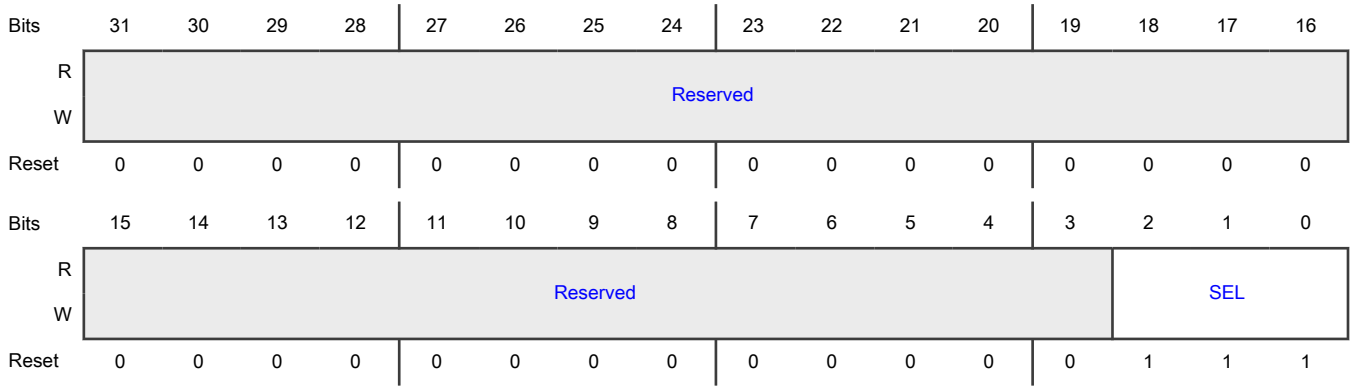
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.67 FLEXIO Clock Selection (FLEXIOCLKSEL)

Offset

Register	Offset
FLEXIOCLKSEL	560h

Diagram



Fields

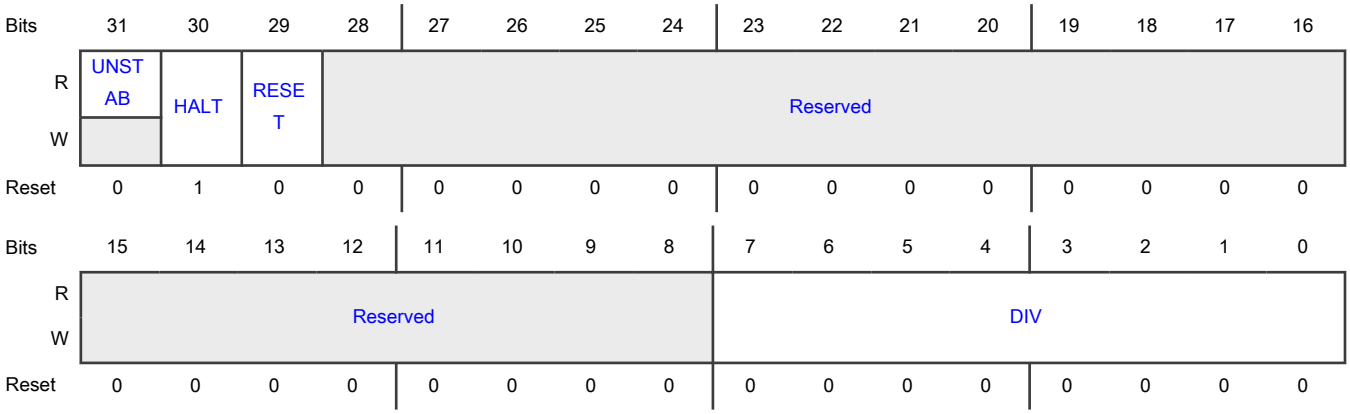
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the FLEXIO clock 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - FRO_12M clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.68 FLEXIO Function Clock Divider (FLEXIOCLKDIV)

Offset

Register	Offset
FLEXIOCLKDIV	564h

Diagram



Fields

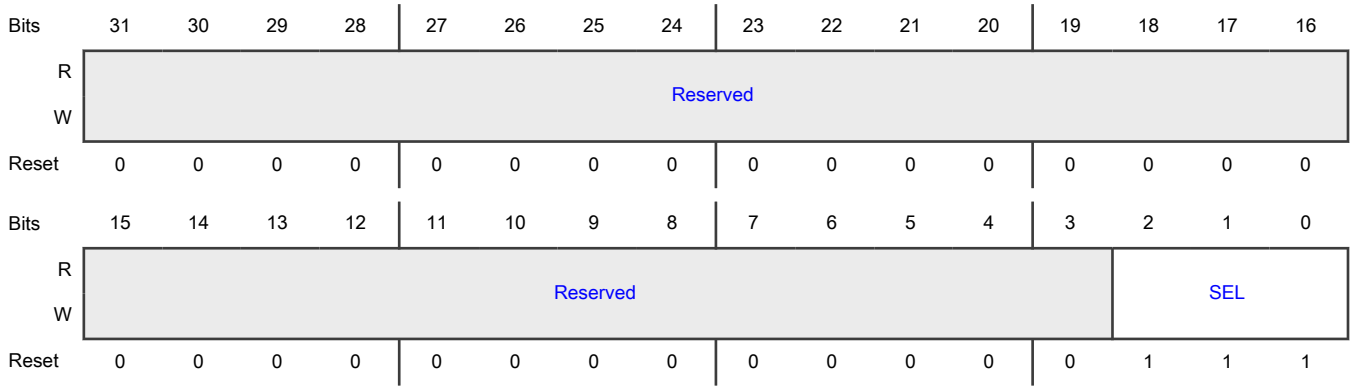
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.69 FLEXCAN0 Clock Selection (FLEXCAN0CLKSEL)

Offset

Register	Offset
FLEXCAN0CLKSEL	5A0h

Diagram



Fields

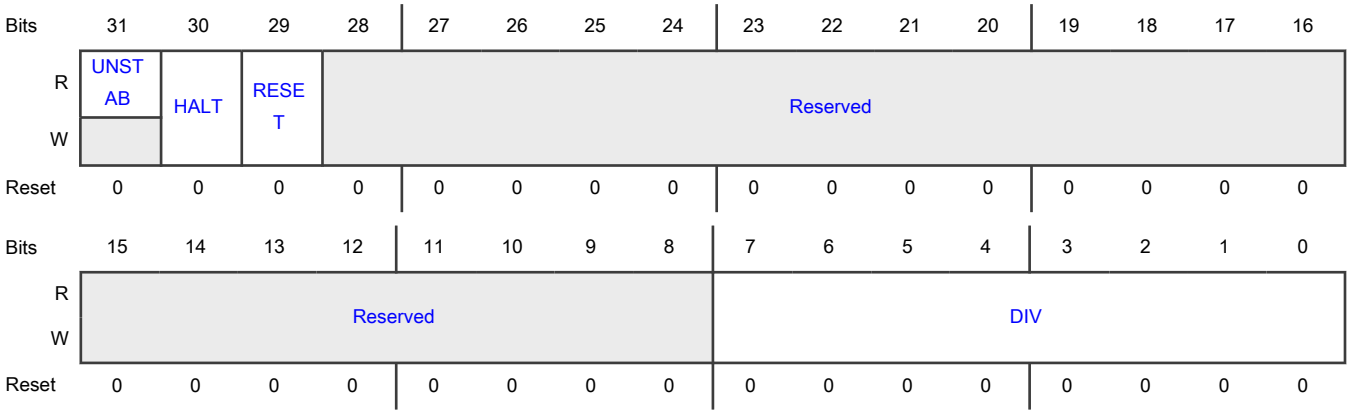
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the FLEXCAN0 clock 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - No clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.70 FLEXCAN0 Function Clock Divider (FLEXCAN0CLKDIV)

Offset

Register	Offset
FLEXCAN0CLKDIV	5A4h

Diagram



Fields

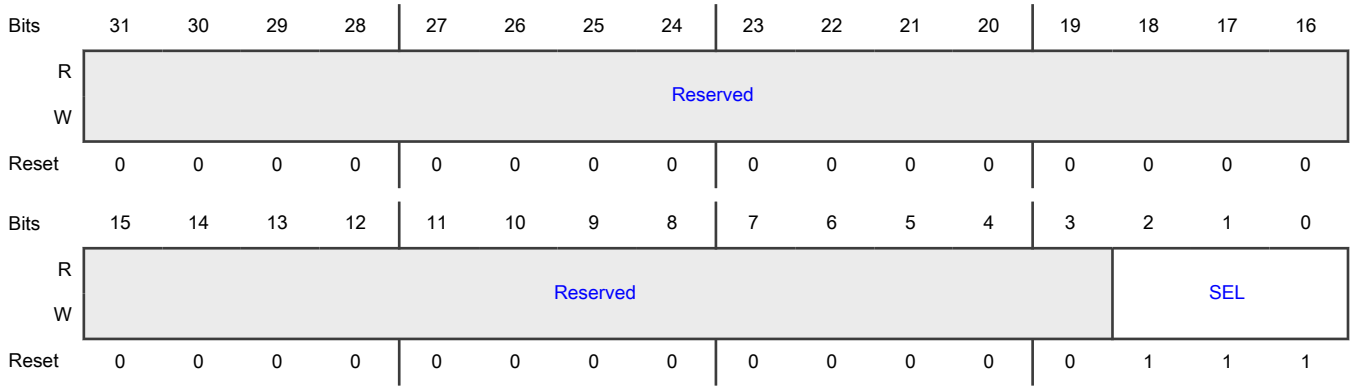
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.71 FLEXCAN1 Clock Selection (FLEXCAN1CLKSEL)

Offset

Register	Offset
FLEXCAN1CLKSEL	5A8h

Diagram



Fields

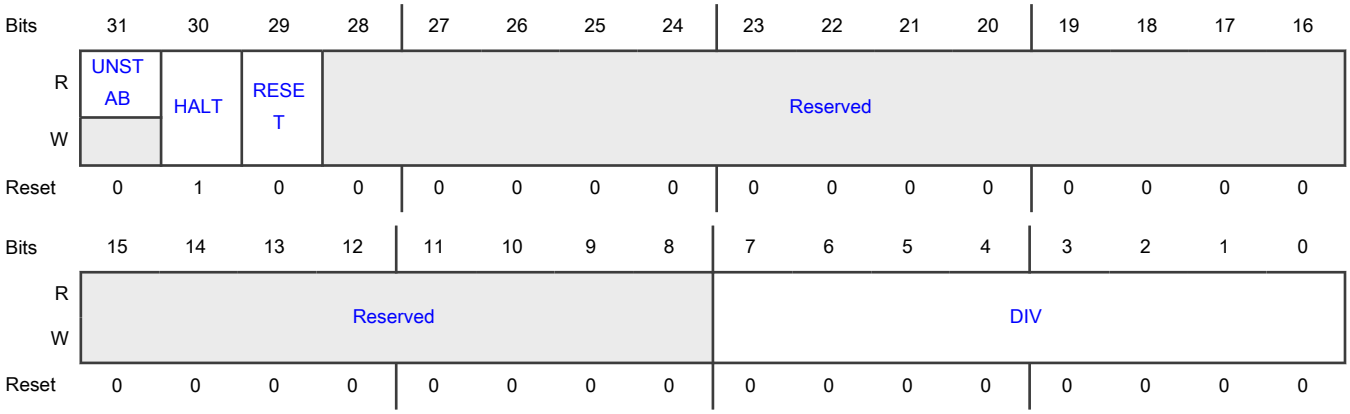
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the FLEXCAN1 clock 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - No clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.72 FLEXCAN1 Function Clock Divider (FLEXCAN1CLKDIV)

Offset

Register	Offset
FLEXCAN1CLKDIV	5ACh

Diagram



Fields

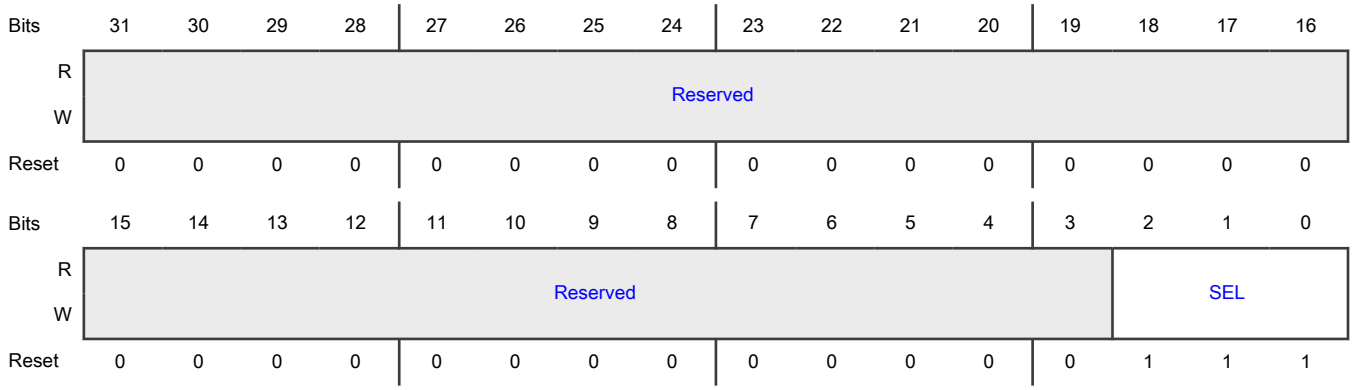
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.73 Ethernet RMII Clock Selection (ENETRMIICLKSEL)

Offset

Register	Offset
ENETRMIICLKSEL	5B0h

Diagram



Fields

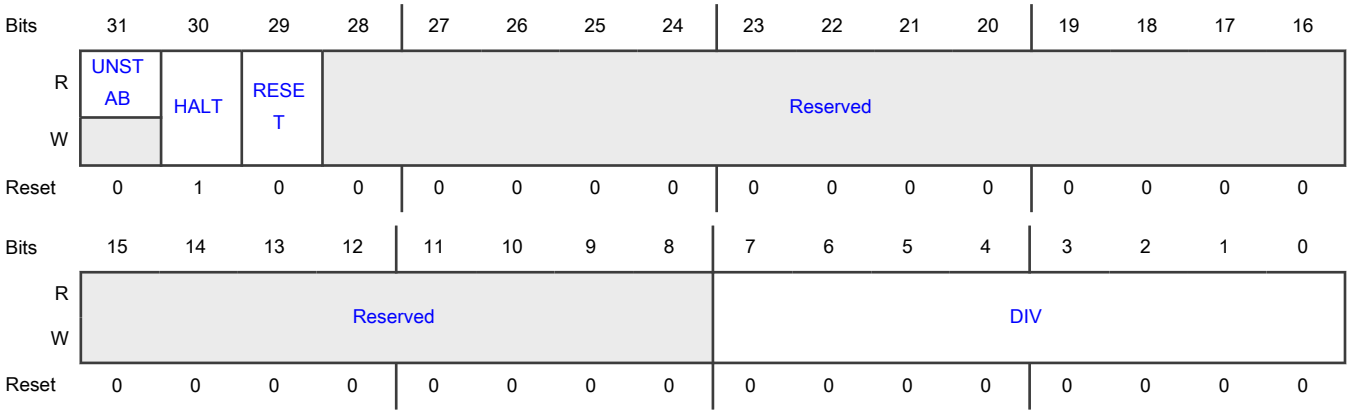
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the Ethernet RMII clock 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - No clock 100b - No clock 101b - PLL1_clk0 clock 110b - No clock 111b - No clock

17.4.1.74 Ethernet RMII Function Clock Divider (ENETRMIICLKDIV)

Offset

Register	Offset
ENETRMIICLKDIV	5B4h

Diagram



Fields

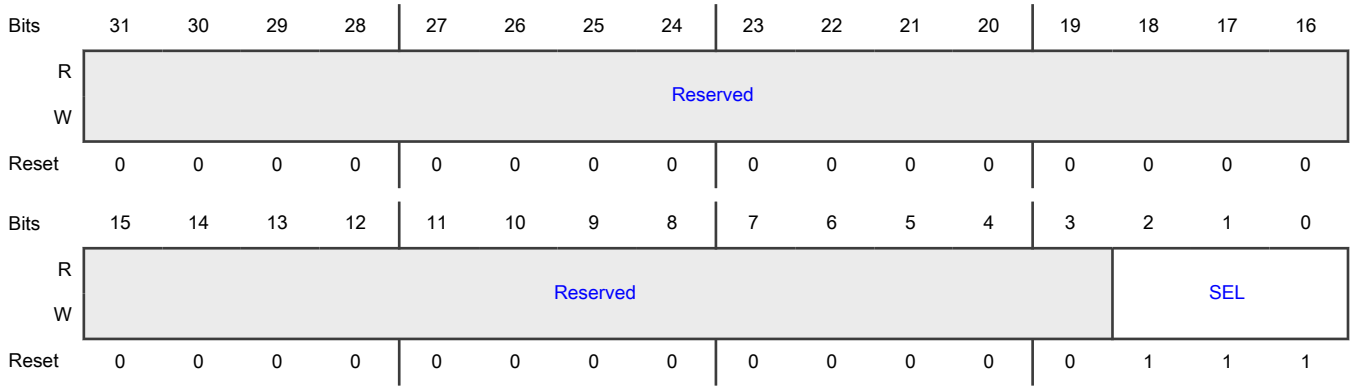
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.75 Ethernet PTP REF Clock Selection (ENETPTPREFCLKSEL)

Offset

Register	Offset
ENETPTPREFCLKSEL	5B8h

Diagram



Fields

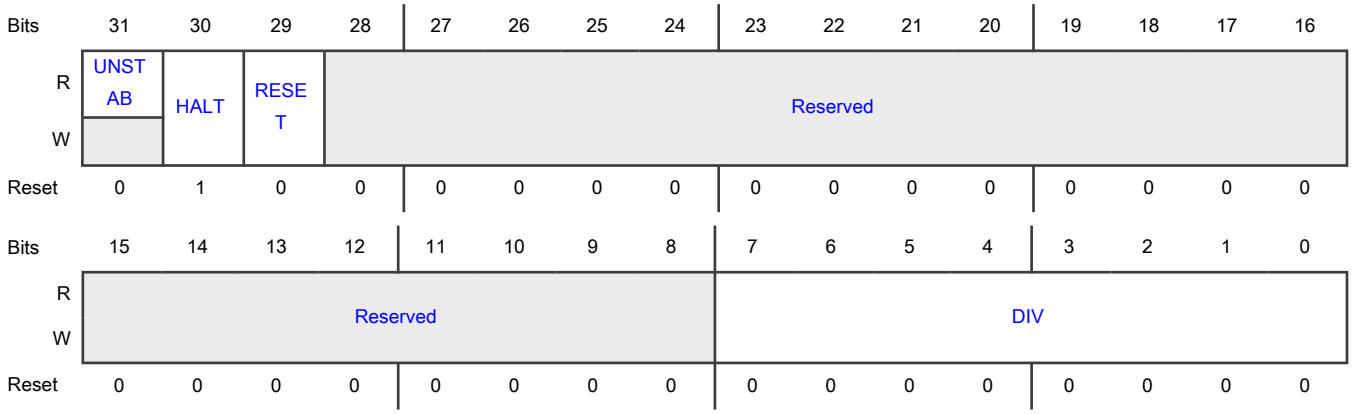
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the Ethernet PTP REF clock 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - No clock 100b - enet0_tx_clk clock 101b - pll1_clk1 clock 110b - No clock 111b - No clock

17.4.1.76 Ethernet PTP REF Function Clock Divider (ENETPTPREFCLKDIV)

Offset

Register	Offset
ENETPTPREFCLKDIV	5BCh

Diagram



Fields

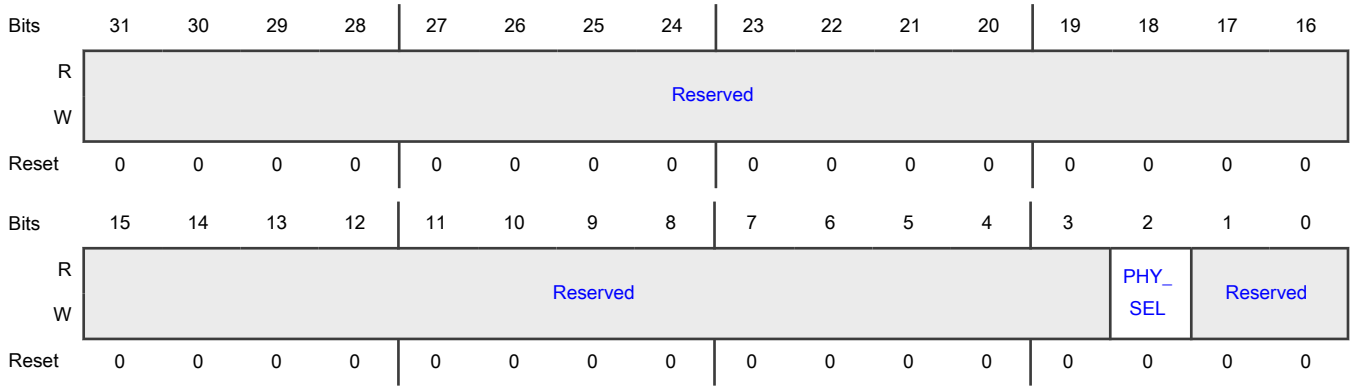
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.77 Ethernet PHY Interface Select (ENET_PHY_INTF_SEL)

Offset

Register	Offset
ENET_PHY_INTF_SEL	5C0h

Diagram



Fields

Field	Function
31-3 —	Reserved
2 PHY_SEL	Selects the PHY interface 0b - Selects MII PHY Interface 1b - Selects RMII PHY Interface
1-0 —	Reserved

17.4.1.78 Sideband Flow Control (ENET_SBD_FLOW_CTRL)

Offset

Register	Offset
ENET_SBD_FLOW_CTRL	5C4h

Function

When set high, instructs the MAC to transmit Pause frames in the Full-duplex mode. In the half-duplex mode, the MAC enables the back-pressure function until this signal is made low again.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														SEL_ch1	SEL_ch0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-2 —	Reserved
1 SEL_ch1	Sideband Flow Control for channel1 0b - No trigger flow control 1b - Trigger flow control
0 SEL_ch0	Sideband Flow Control for channel0 0b - No trigger flow control 1b - Trigger flow control

17.4.1.79 EWM0 Clock Selection (EWM0CLKSEL)

Offset

Register	Offset
EWM0CLKSEL	5D4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															SEL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

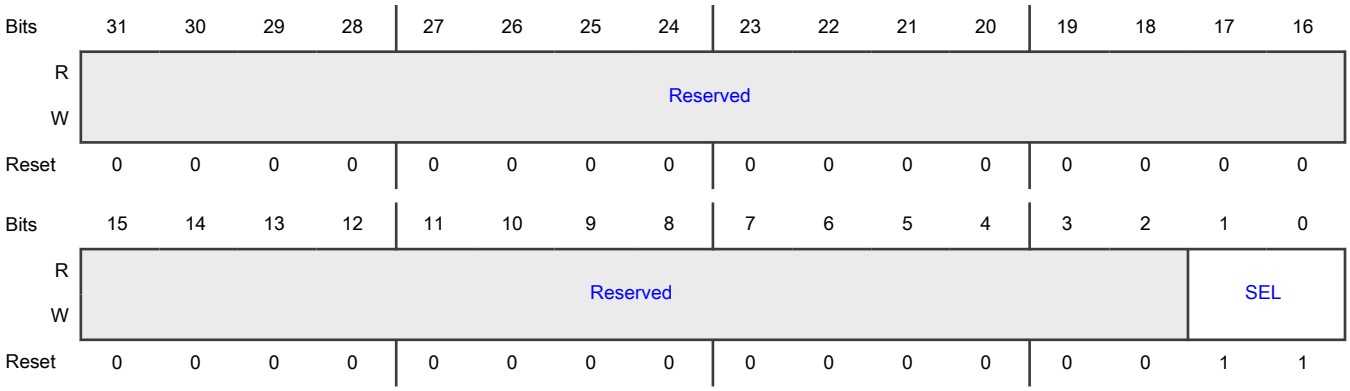
Field	Function
31-1 —	Reserved
0 SEL	Selects the EWM0 clock 0b - clk_16k[2] 1b - xtal32k[2]

17.4.1.80 WDT1 Clock Selection (WDT1CLKSEL)

Offset

Register	Offset
WDT1CLKSEL	5D8h

Diagram



Fields

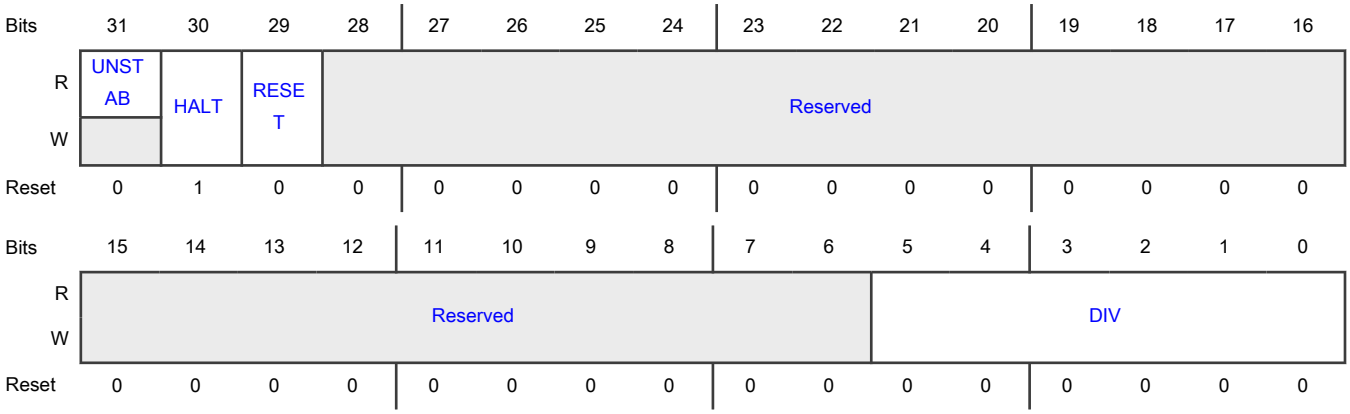
Field	Function
31-2 —	Reserved
1-0 SEL	Selects the WDT1 clock 00b - FRO16K clock 2 01b - fro_hf_div clock 10b - clk_1m clock 11b - clk_1m clock

17.4.1.81 WDT1 Function Clock Divider (WDT1CLKDIV)

Offset

Register	Offset
WDT1CLKDIV	5DCh

Diagram



Fields

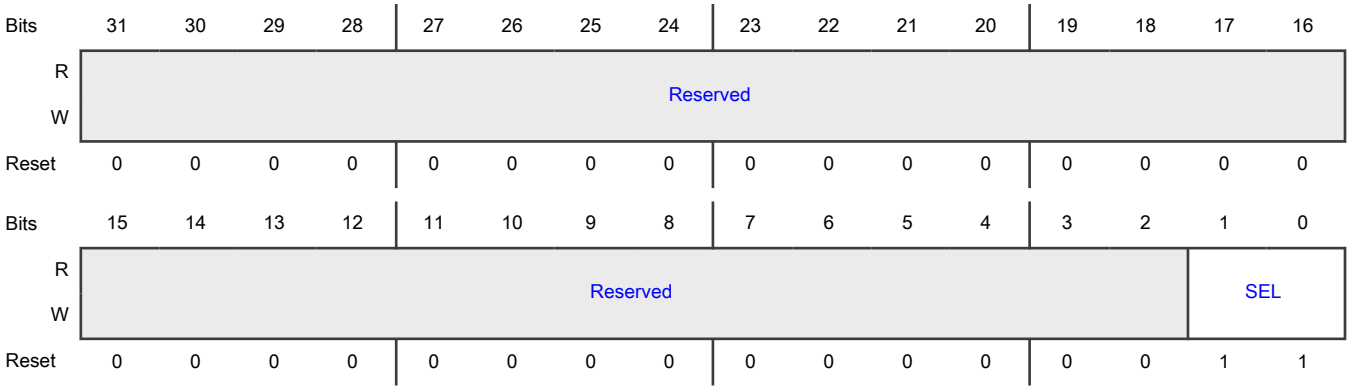
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-6 —	Reserved
5-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.82 OSTIMER Clock Selection (OSTIMERCLKSEL)

Offset

Register	Offset
OSTIMERCLKSEL	5E0h

Diagram



Fields

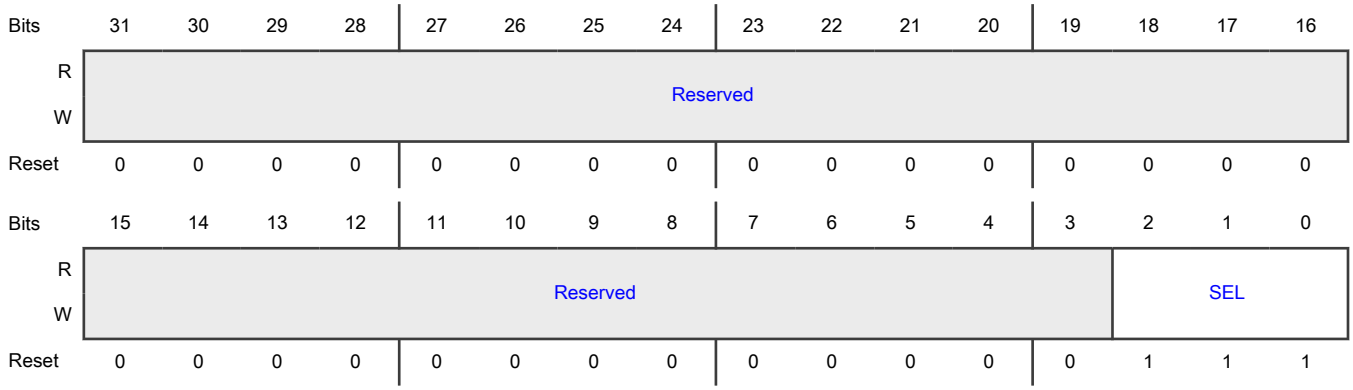
Field	Function
31-2 —	Reserved
1-0 SEL	Selects the OS Event Timer clock 00b - clk_16k[2] 01b - xtal32k[2] 10b - clk_1m clock 11b - No clock

17.4.1.83 CMP0 Function Clock Selection (CMP0FCLKSEL)

Offset

Register	Offset
CMP0FCLKSEL	5F0h

Diagram



Fields

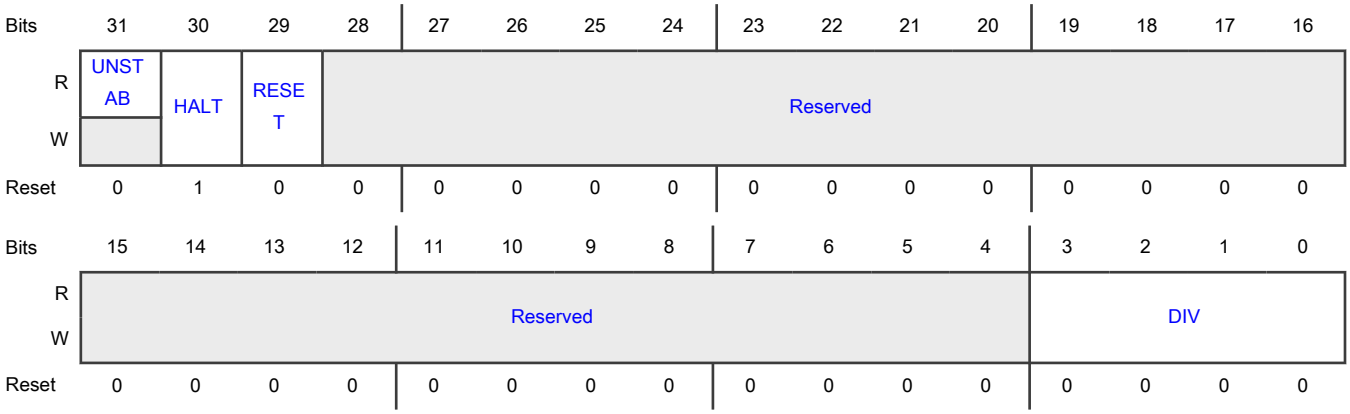
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the CMP0 function clock 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO_12M clock 100b - CLKIN clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.84 CMP0 Function Clock Divider (CMP0FCLKDIV)

Offset

Register	Offset
CMP0FCLKDIV	5F4h

Diagram



Fields

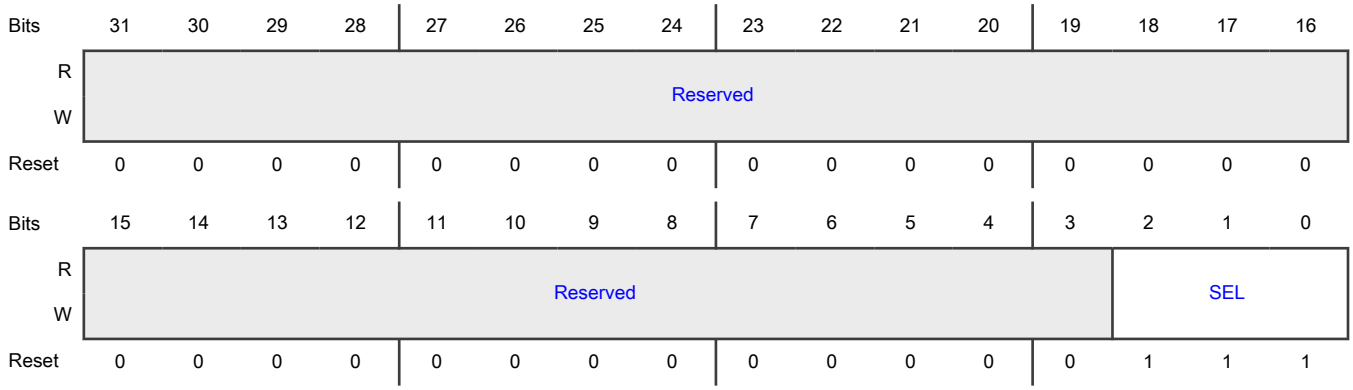
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-4 —	Reserved
3-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.85 CMP0 Round Robin Clock Selection (CMP0RRCLKSEL)

Offset

Register	Offset
CMP0RRCLKSEL	5F8h

Diagram



Fields

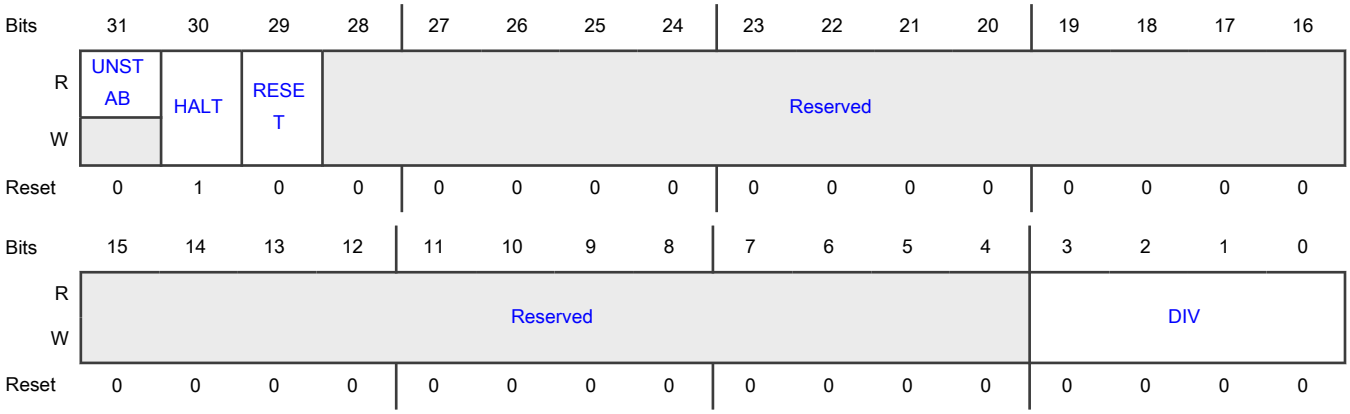
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the CMP0 round robin clock 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO_12M clock 100b - CLKIN clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.86 CMP0 Round Robin Clock Divider (CMP0RRCLKDIV)

Offset

Register	Offset
CMP0RRCLKDIV	5FCh

Diagram



Fields

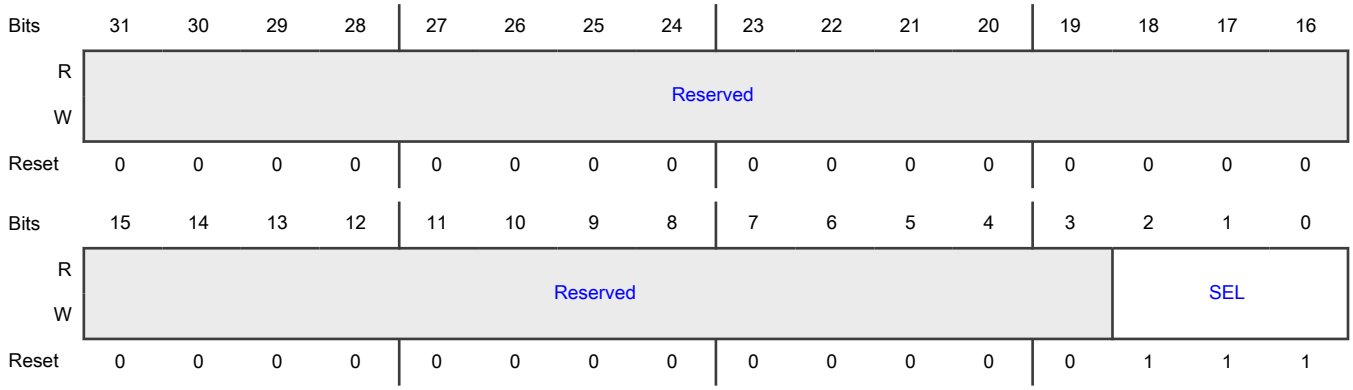
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-4 —	Reserved
3-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.87 CMP1 Function Clock Selection (CMP1FCLKSEL)

Offset

Register	Offset
CMP1FCLKSEL	600h

Diagram



Fields

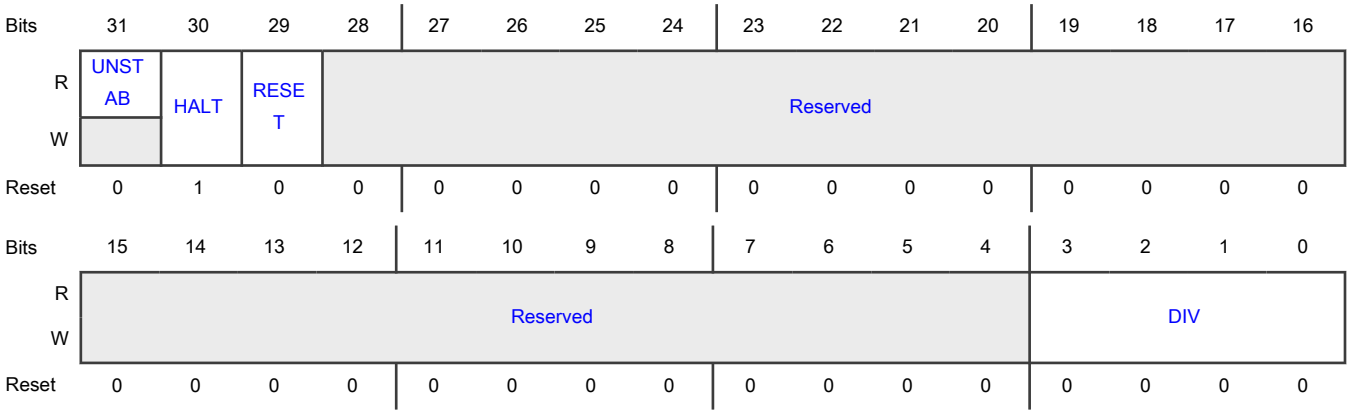
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the CMP1 function clock 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO_12M clock 100b - CLKIN clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.88 CMP1 Function Clock Divider (CMP1FCLKDIV)

Offset

Register	Offset
CMP1FCLKDIV	604h

Diagram



Fields

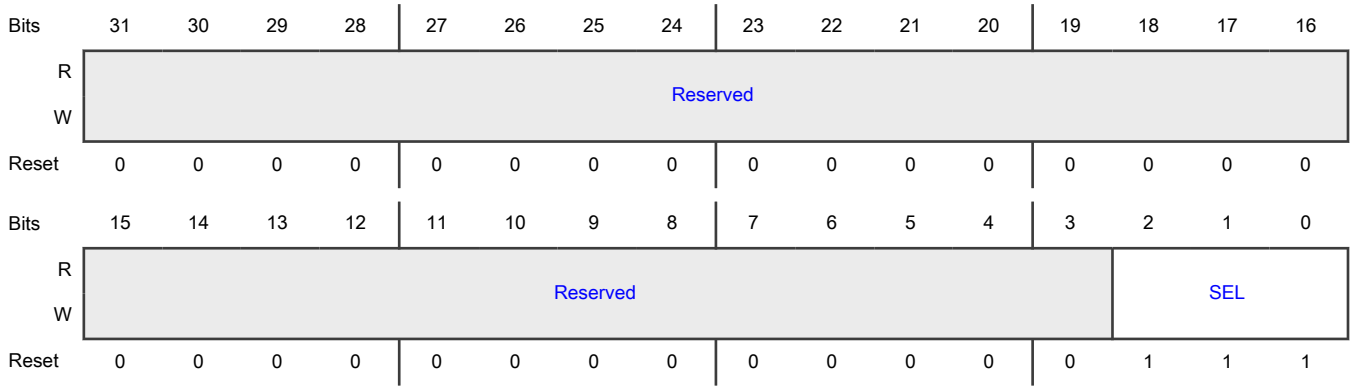
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-4 —	Reserved
3-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.89 CMP1 Round Robin Clock Source Select (CMP1RRCLKSEL)

Offset

Register	Offset
CMP1RRCLKSEL	608h

Diagram



Fields

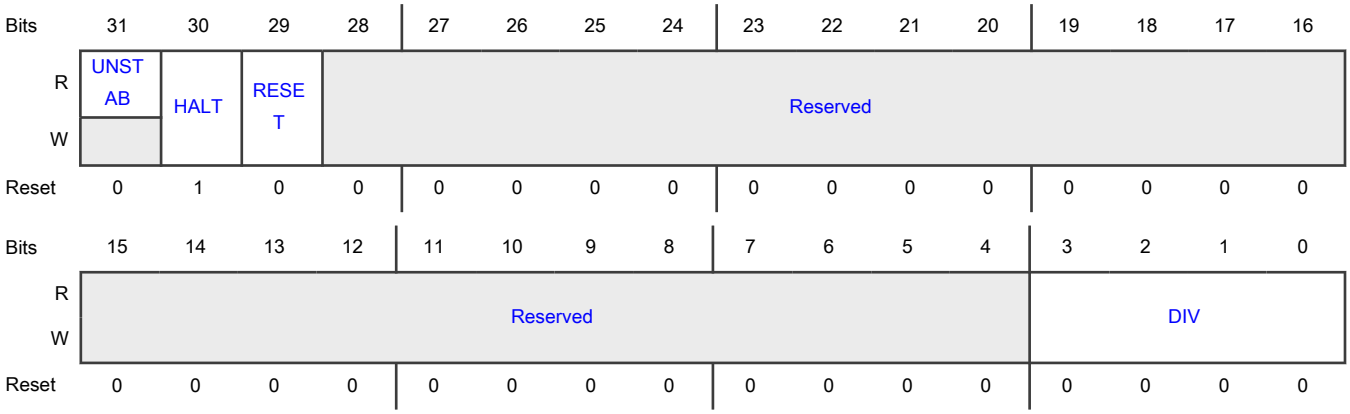
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the CMP1 round robin clock 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO_12M clock 100b - CLKIN clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.90 CMP1 Round Robin Clock Division (CMP1RRCLKDIV)

Offset

Register	Offset
CMP1RRCLKDIV	60Ch

Diagram



Fields

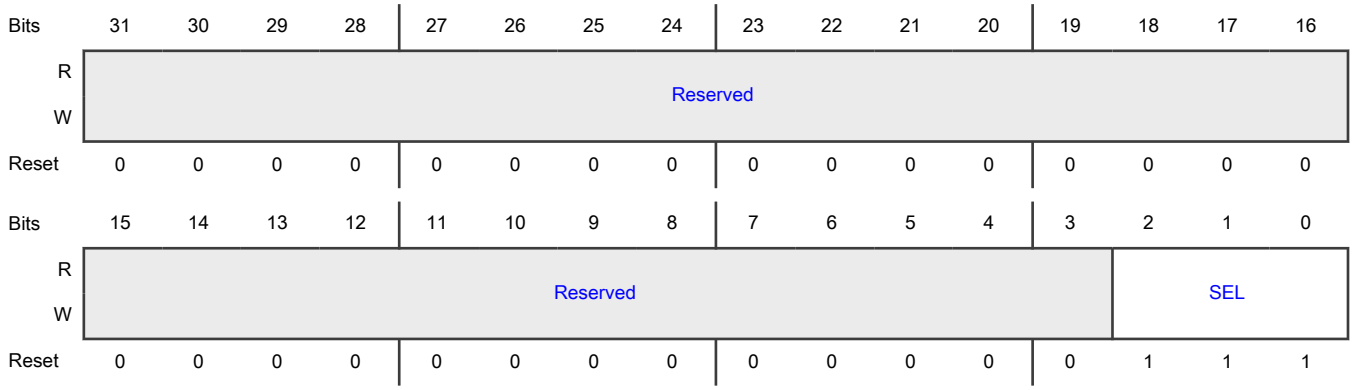
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-4 —	Reserved
3-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.91 CMP2 Function Clock Source Select (CMP2FCLKSEL)

Offset

Register	Offset
CMP2FCLKSEL	610h

Diagram



Fields

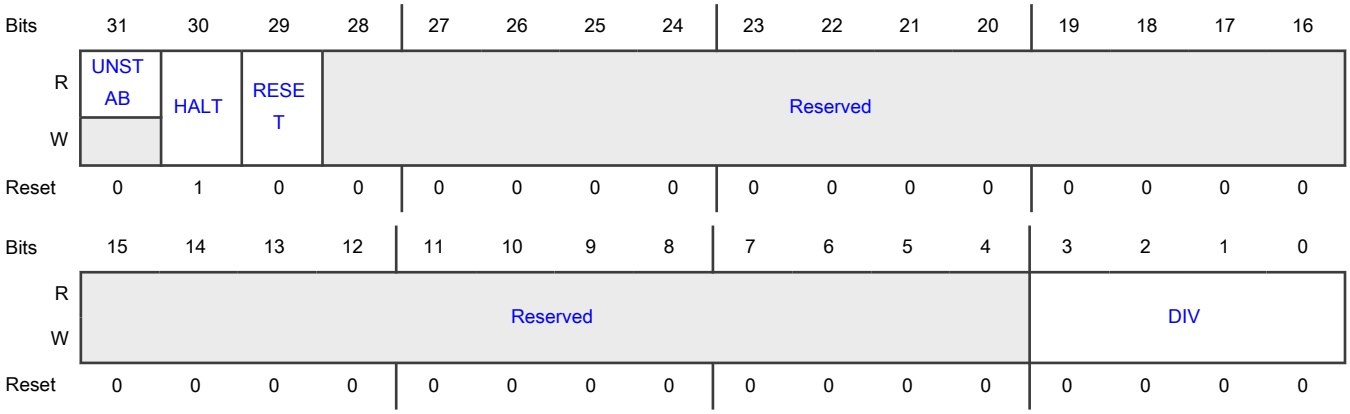
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the CMP2 function clock 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO_12M clock 100b - CLKIN clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.92 CMP2 Function Clock Division (CMP2FCLKDIV)

Offset

Register	Offset
CMP2FCLKDIV	614h

Diagram



Fields

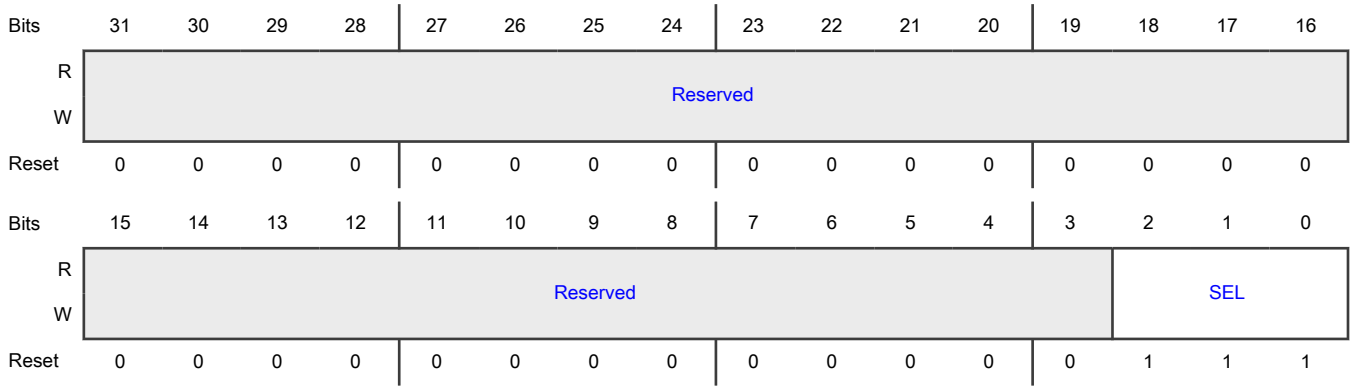
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-4 —	Reserved
3-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.93 CMP2 Round Robin Clock Source Select (CMP2RRCLKSEL)

Offset

Register	Offset
CMP2RRCLKSEL	618h

Diagram



Fields

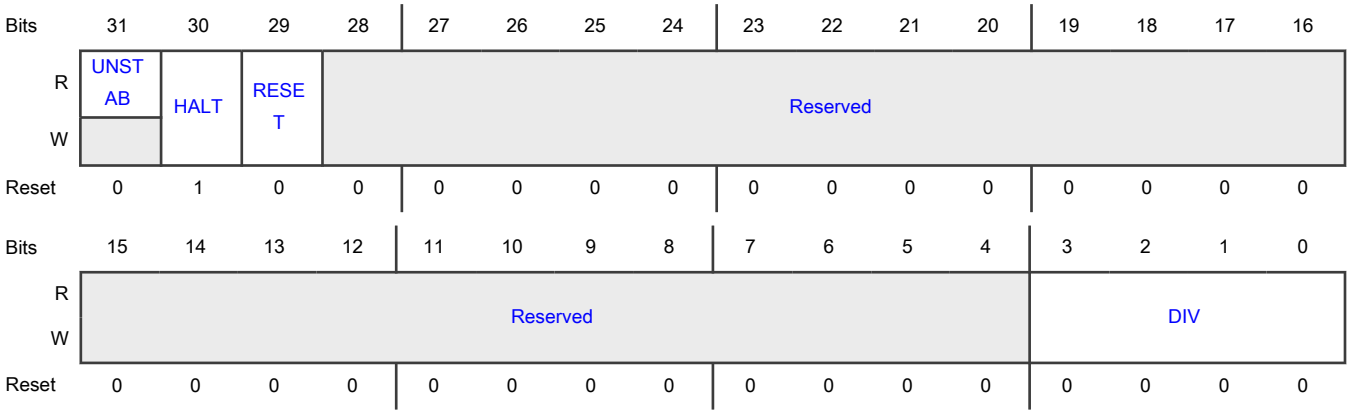
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the CMP2 round robin clock 000b - No clock 001b - PLL0 clock 010b - FRO_HF clock 011b - FRO_12M clock 100b - CLKIN clock 101b - PLL1_clk0 clock0 110b - USB PLL clock 111b - No clock

17.4.1.94 CMP2 Round Robin Clock Division (CMP2RRCLKDIV)

Offset

Register	Offset
CMP2RRCLKDIV	61Ch

Diagram



Fields

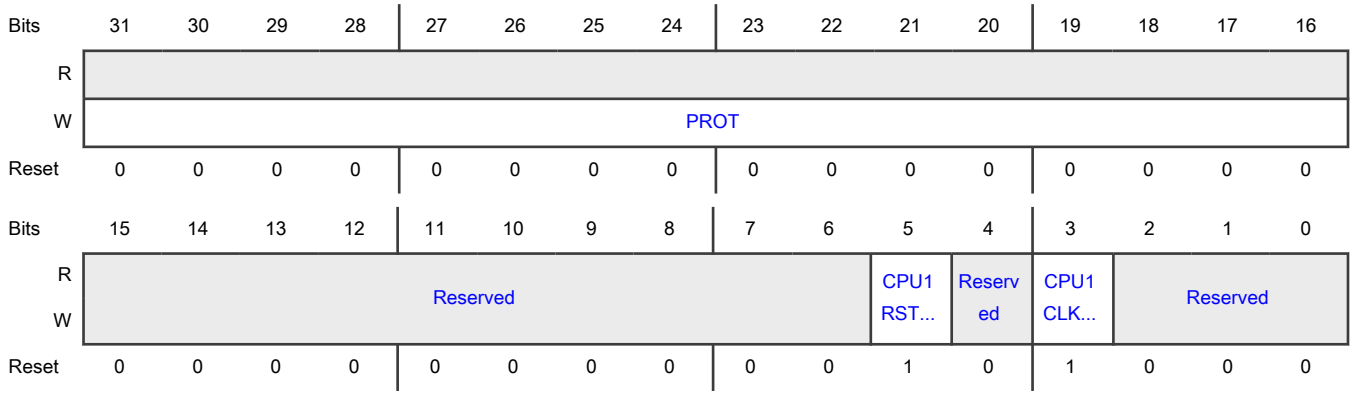
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-4 —	Reserved
3-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.95 CPU Control for Multiple Processors (CPUCTRL)

Offset

Register	Offset
CPUCTRL	800h

Diagram



Fields

Field	Function
31-16 PROT	Write Protect Must be written as 0xC0C4 for the write to have an effect 1100_0000_1100_0100b - For write operation to have an effect.
15-6 —	Reserved
5 CPU1RSTEN	CPU1 reset Controls the reset signal to CPU1. This bit can be cleared to release CPU1 from reset after a valid boot address has been written to CPBOOT. 0b - The CPU1 is not reset. 1b - The CPU1 is reset.
4 —	Reserved
3 CPU1CLKEN	Enables the CPU1 clock 0b - The CPU1 clock is not enabled 1b - The CPU1 clock is enabled
2-0 —	Reserved

17.4.1.96 Coprocessor Boot Address (CPBOOT)

Offset

Register	Offset
CPBOOT	804h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CPBOOT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CPBOOT								Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-7 CPBOOT	Coprocessor Boot VTOR Address [31:7] for CPU1
6-0 —	Reserved

17.4.1.97 CPU Status (CPUSTAT)

Offset

Register	Offset
CPUSTAT	80Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												CPU1L OC...	CPU0L OC...	CPU1 SLE...	CPU0 SLE...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0	0

Fields

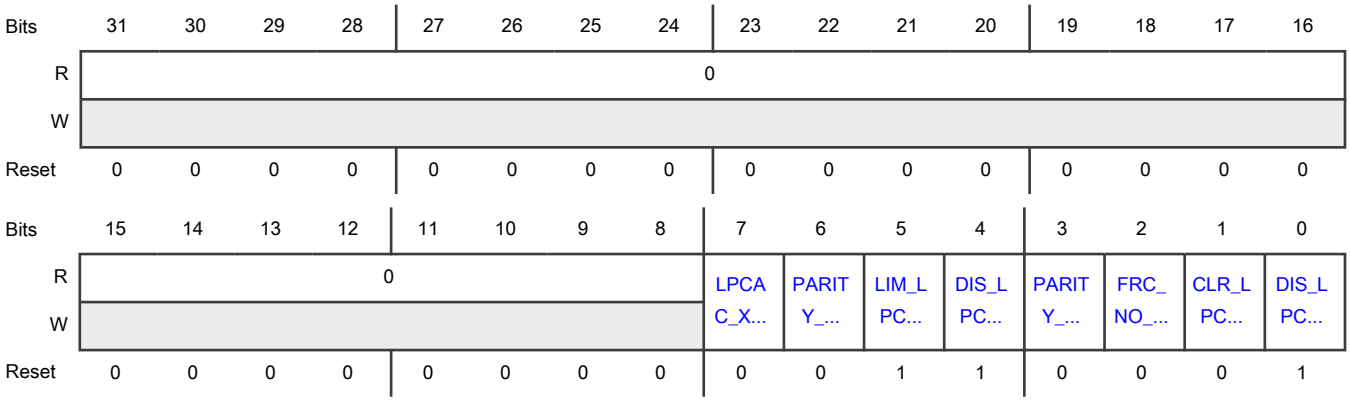
Field	Function
31-4 —	Reserved
3 CPU1LOCKUP	CPU1 lockup state 0b - CPU is not in lockup 1b - CPU is in lockup
2 CPU0LOCKUP	CPU0 lockup state 0b - CPU is not in lockup 1b - CPU is in lockup
1 CPU1SLEEPING	CPU1 sleeping state 0b - CPU is not sleeping 1b - CPU is sleeping
0 CPU0SLEEPING	CPU0 sleeping state 0b - CPU is not sleeping 1b - CPU is sleeping

17.4.1.98 LPCAC Control (LPCAC_CTRL)

Offset

Register	Offset
LPCAC_CTRL	824h

Diagram



Fields

Field	Function
31-8 —	Reserved
7 LPCAC_XOM	LPCAC XOM(eXecute-Only-Memory) attribute control Controls if the instruction fetch attribute is used as part of the address input to the LPCAC. When XOM regions in the internal flash are not configured at the MBC, then this option should be disabled so that instructions and data can be stored within the same cache line. This provides the best cache efficiency for non-XOM applications. When XOM areas in the internal flash are configured at the MBC, then this bit must be set so that instructions and data are cached using separate lines within the LPCAC. 0b - Disabled. 1b - Enabled.
6 PARITY_FAULT_EN	Enable parity error report. 0b - Disables parity error report 1b - Enables parity error report
5 LIM_LPCAC_WTBF	Limit LPCAC Write Through Buffer. 0b - Write buffer enabled when transaction is bufferable. 1b - Write buffer enabled when transaction is cacheable and bufferable
4 DIS_LPCAC_WTBF	Disable LPCAC Write Through Buffer. 0b - Enables write through buffer 1b - Disables write through buffer
3 PARITY_MISS_EN	Enables parity miss. 0b - Disabled 1b - Enables parity, miss on parity error
2 FRC_NO_ALLOC	Forces no allocation. 0b - Forces allocation 1b - Forces no allocation
1 CLR_LPCAC	Clears the cache function. 0b - Unclears the cache 1b - Clears the cache
0 DIS_LPCAC	Disables/enables the cache function. 0b - Enabled 1b - Disabled

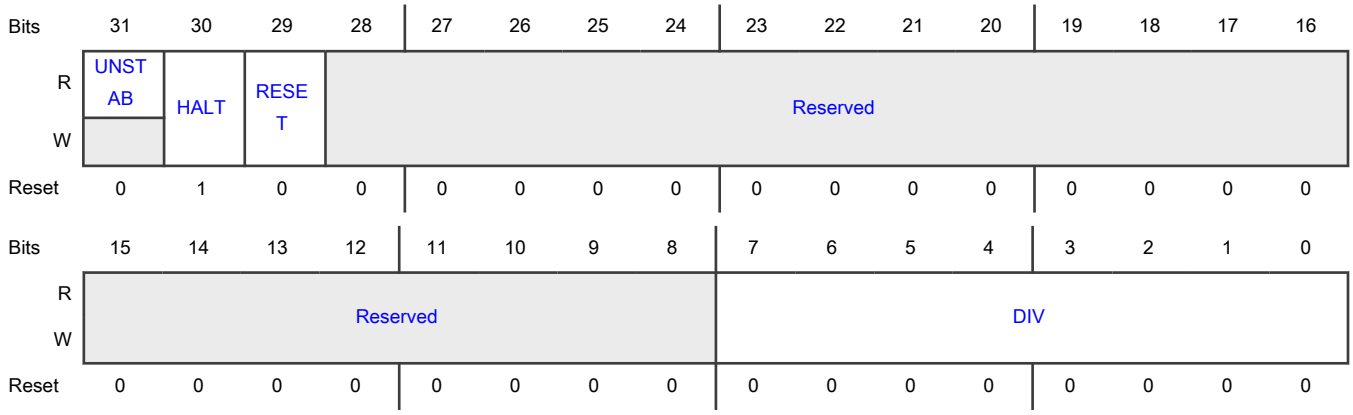
17.4.1.99 LP_FLEXCOMM Clock Divider (FLEXCOMM0CLKDIV - FLEXCOMM9CLKDIV)

Offset

For n = 0 to 9:

Register	Offset
FLEXCOMMnCLKDIV	850h + (n × 4h)

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value • 0 - Divide by 1 • 1 - Divide by 2 • ... • value - Divide by (DIV+1)

17.4.1.100 SAI0 Function Clock Source Select (SAI0CLKSEL)

Offset

Register	Offset
SAI0CLKSEL	880h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved													SEL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

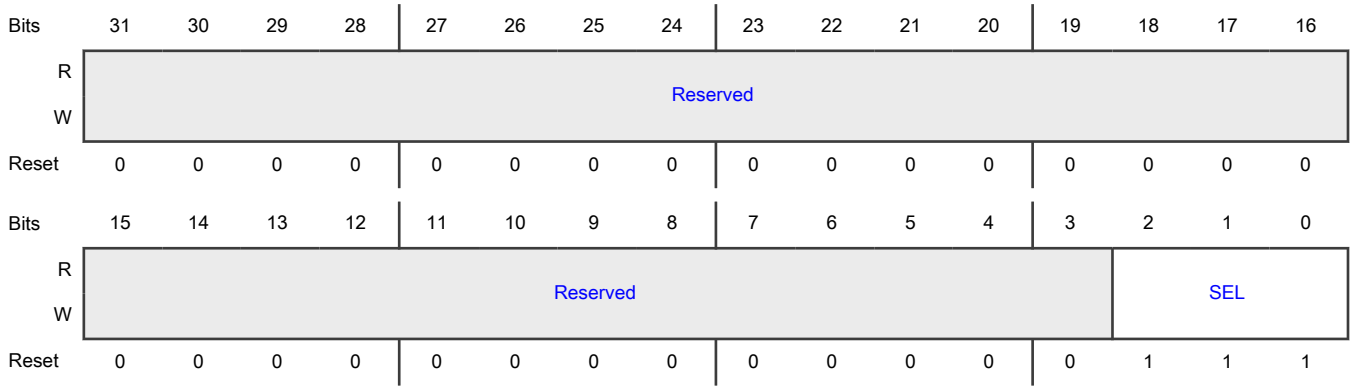
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the clock source 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - PLL1_CLK0 clock 101b - No clock 110b - USB PLL clock 111b - No clock

17.4.1.101 SAI1 Function Clock Source Select (SAI1CLKSEL)

Offset

Register	Offset
SAI1CLKSEL	884h

Diagram



Fields

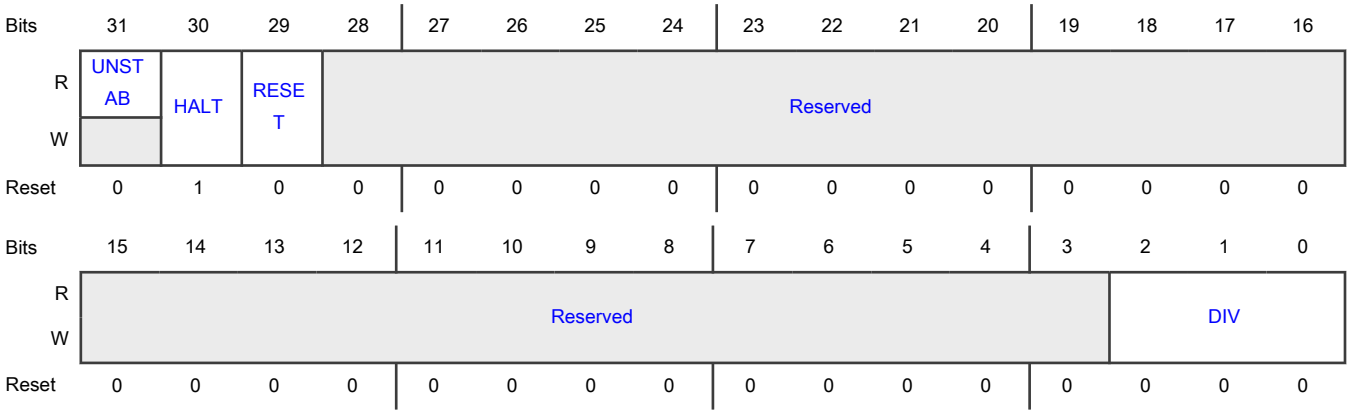
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the clock source 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - PLL1_CLK0 clock 101b - No clock 110b - USB PLL clock 111b - No clock

17.4.1.102 SAI0 Function Clock Division (SAI0CLKDIV)

Offset

Register	Offset
SAI0CLKDIV	888h

Diagram



Fields

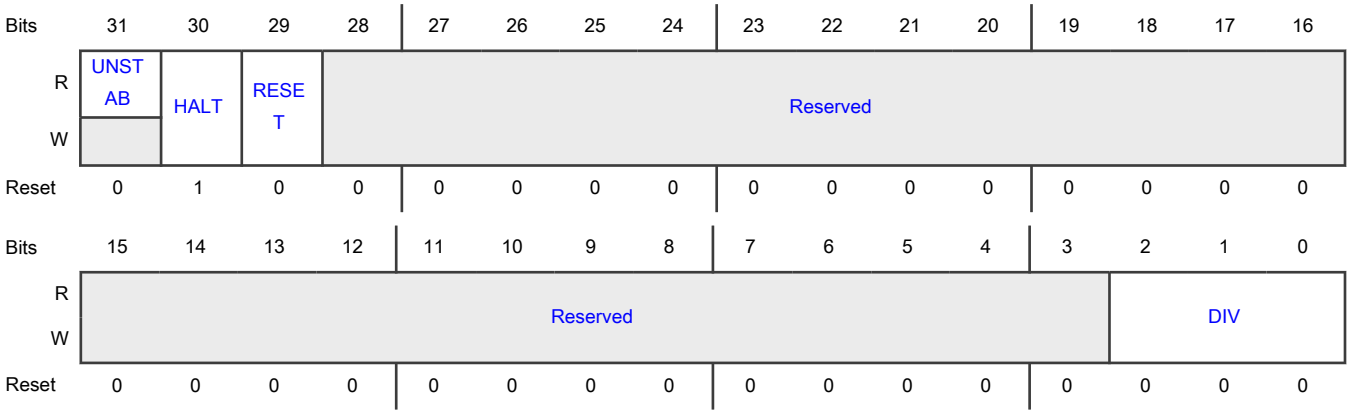
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.103 SAI1 Function Clock Division (SAI1CLKDIV)

Offset

Register	Offset
SAI1CLKDIV	88Ch

Diagram



Fields

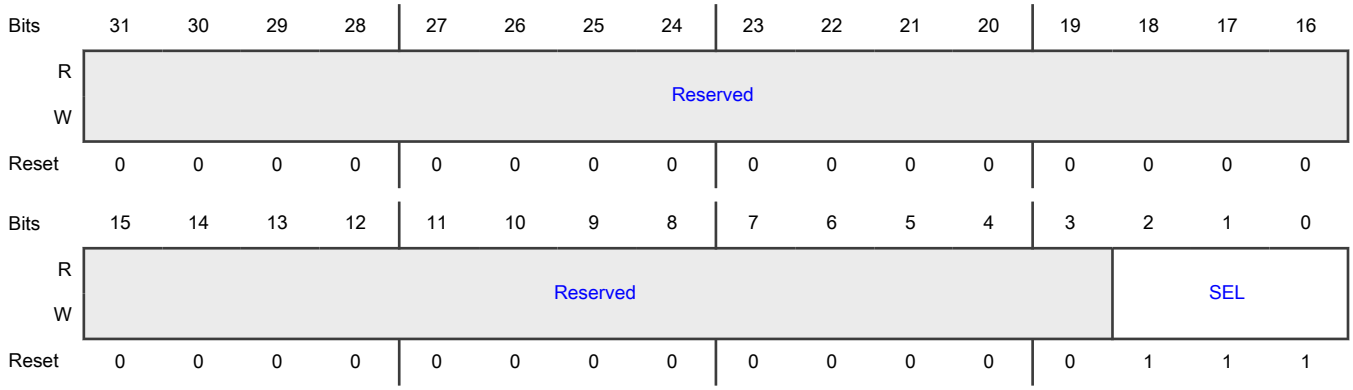
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1).

17.4.1.104 EMVSIM0 Clock Source Select (EMVSIM0CLKSEL)

Offset

Register	Offset
EMVSIM0CLKSEL	890h

Diagram



Fields

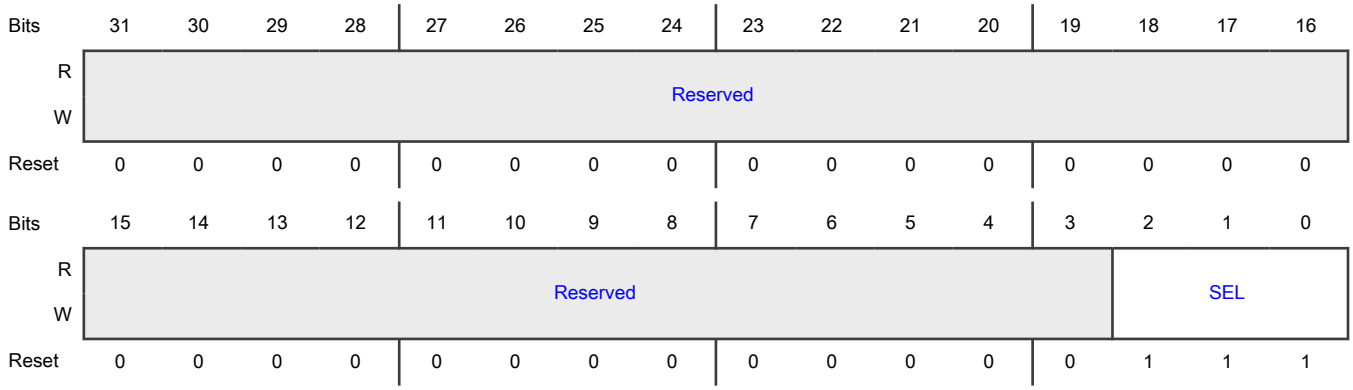
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the EMVSIM0 function clock source 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - FRO_12M clock 101b - PLL1_clk0 clock0 110b - No clock 111b - No clock

17.4.1.105 EMVSIM1 Clock Source Select (EMVSIM1CLKSEL)

Offset

Register	Offset
EMVSIM1CLKSEL	894h

Diagram



Fields

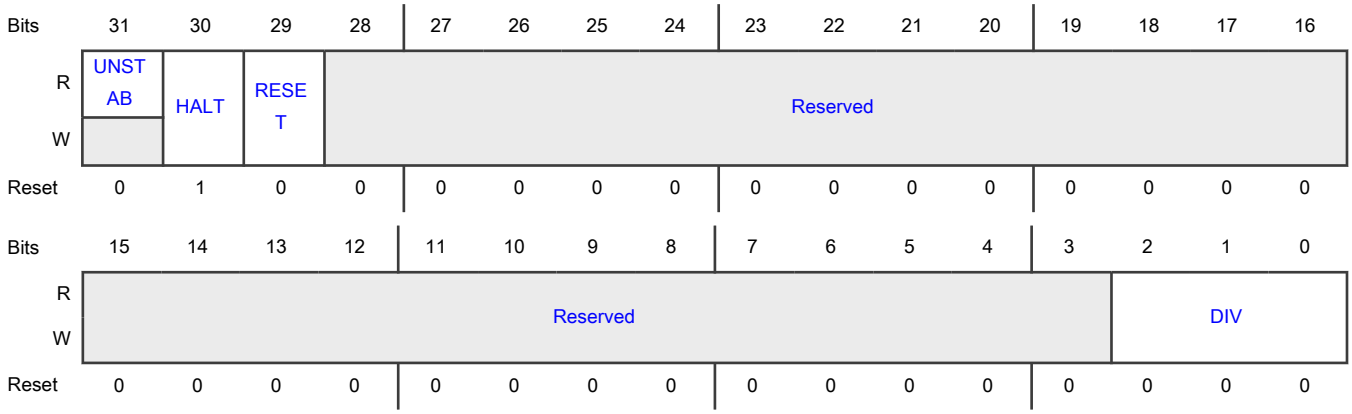
Field	Function
31-3 —	Reserved
2-0 SEL	Selects the EMVSIM1 function clock source 000b - No clock 001b - PLL0 clock 010b - CLKIN clock 011b - FRO_HF clock 100b - FRO_12M clock 101b - PLL1_clk0 clock0 110b - No clock 111b - No clock

17.4.1.106 EMVSIM0 Function Clock Division (EMVSIM0CLKDIV)

Offset

Register	Offset
EMVSIM0CLKDIV	898h

Diagram



Fields

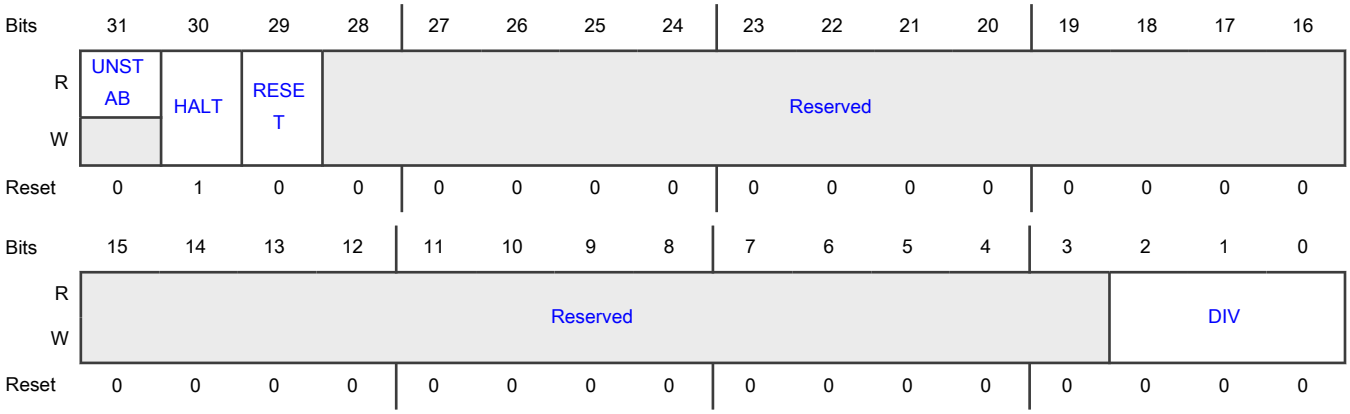
Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.107 EMVSIM1 Function Clock Division (EMVSIM1CLKDIV)

Offset

Register	Offset
EMVSIM1CLKDIV	89Ch

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-3 —	Reserved
2-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.108 Clock Control (CLOCK_CTRL)

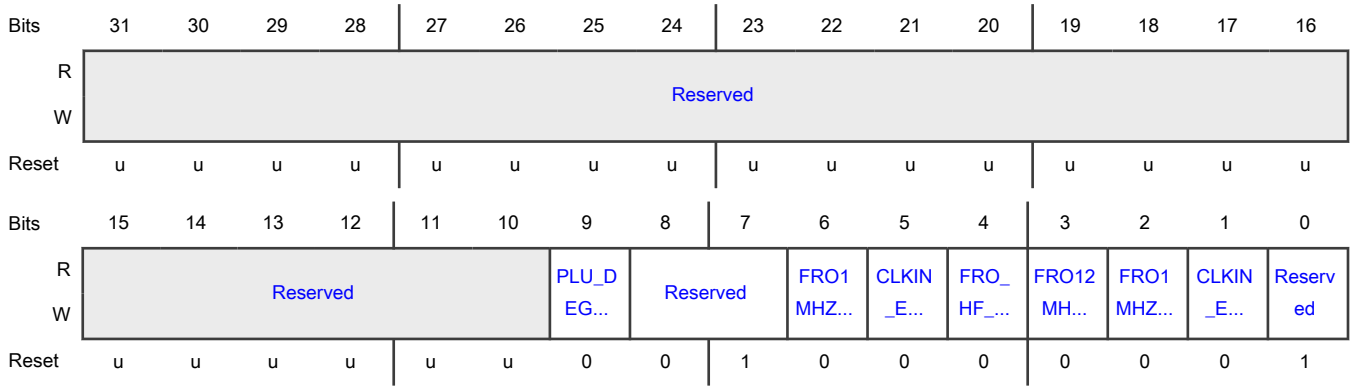
Offset

Register	Offset
CLOCK_CTRL	A18h

Function

Various system clocks enable controls

Diagram



Fields

Field	Function
31-10 —	Reserved
9 PLU_DEGLITCH_CLK_ENA	Enables clocks FRO_1MHz and FRO_12MHz for PLU deglitching. 0b - Clock is not enabled 1b - Clock is enabled
8-7 —	Reserved
6 FRO1MHZ_CLK_ENA	Enables FRO_1MHz clock for clock muxing in clock gen 0b - Clock is not enabled 1b - Clock is enabled
5 CLKIN_ENA	Enables clk_in clock for MICFIL, EMVSIM0/1, CAN0/1, I3C0/1, SAI0/1, SINC Filter (SINC), TSI, USBFS, SCT, uSDHC, clkout. 0b - Clock is not enabled 1b - Clock is enabled
4 FRO_HF_ENA	Enables FRO HF clock for the Frequency Measure module 0b - Clock is not enabled 1b - Clock is enabled
3 FRO12MHZ_ENA	Enables the FRO_12MHz clock for the Flash, LPTMR0/1, and Frequency Measurement modules 0b - Clock is not enabled 1b - Clock is enabled
2	Enables the FRO_1MHz clock for RTC module and for UTICK 0b - Clock is not enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
FRO1MHZ_ENA	1b - Clock is enabled
1 CLKIN_ENA_FM_USBH_LPT	Enables the clk_in clock for the Frequency Measurement, USB HS and LPTMR0/1 modules. 0b - Clock is not enabled 1b - Clock is enabled
0 —	Reserved

17.4.1.109 I3C1 Functional Clock Selection (I3C1FCLKSEL)

Offset

Register	Offset
I3C1FCLKSEL	B30h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														SEL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	I3C1 clock select 000b - No clock 001b - PLL0 clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	010b - CLKIN clock 011b - FRO_HF clock 100b - No clock 101b - PLL1_clk0 clock 110b - USB PLL clock 111b - No clock

17.4.1.110 Selects the I3C1 Time Control clock (I3C1FCLKSTCSEL)

Offset

Register	Offset
I3C1FCLKSTCSEL	B34h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved													SEL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	I3C1 FCLK_STC clock select 000b - I3C1 functional clock I3C1FCLK 001b - FRO_1M clock 010b - No clock

Table continues on the next page...

Table continued from the previous page...

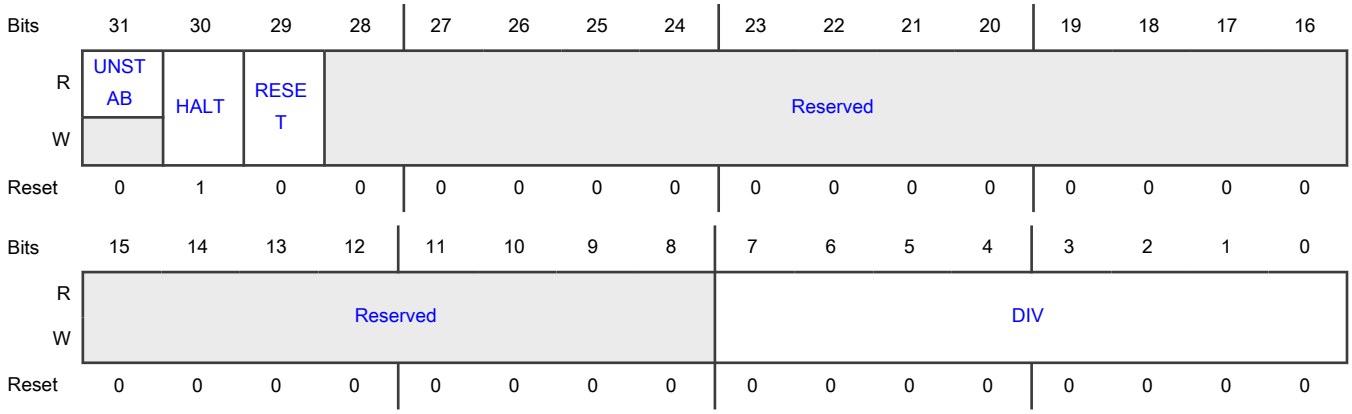
Field	Function
	011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.111 I3C1 FCLK_STC Clock Divider (I3C1FCLKSTCDIV)

Offset

Register	Offset
I3C1FCLKSTCDIV	B38h

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped
29	Resets the divider counter

Table continues on the next page...

Table continued from the previous page...

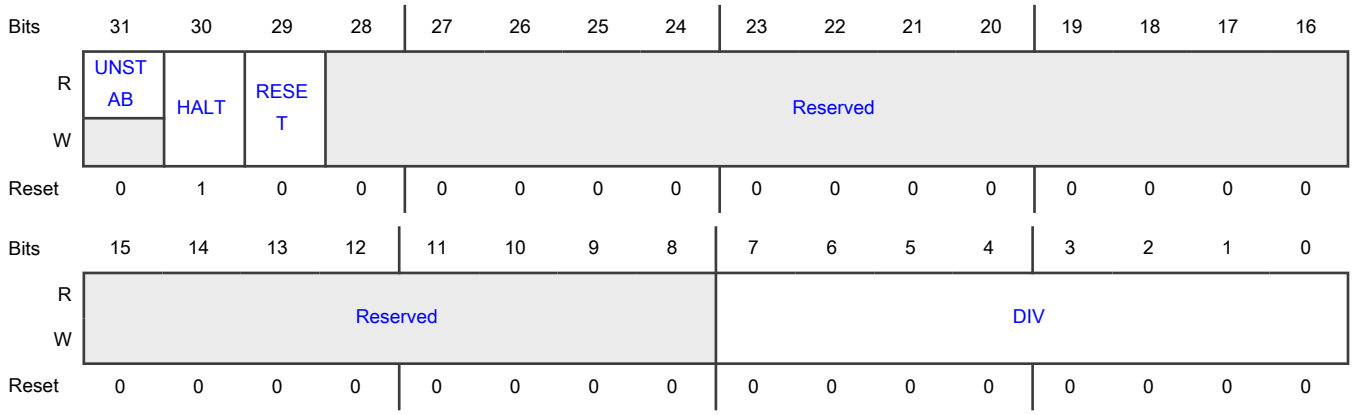
Field	Function
RESET	0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.112 I3C1 FCLK Slow clock Divider (I3C1FCLKSDIV)

Offset

Register	Offset
I3C1FCLKSDIV	B3Ch

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running 1b - Divider clock is stopped

Table continues on the next page...

Table continued from the previous page...

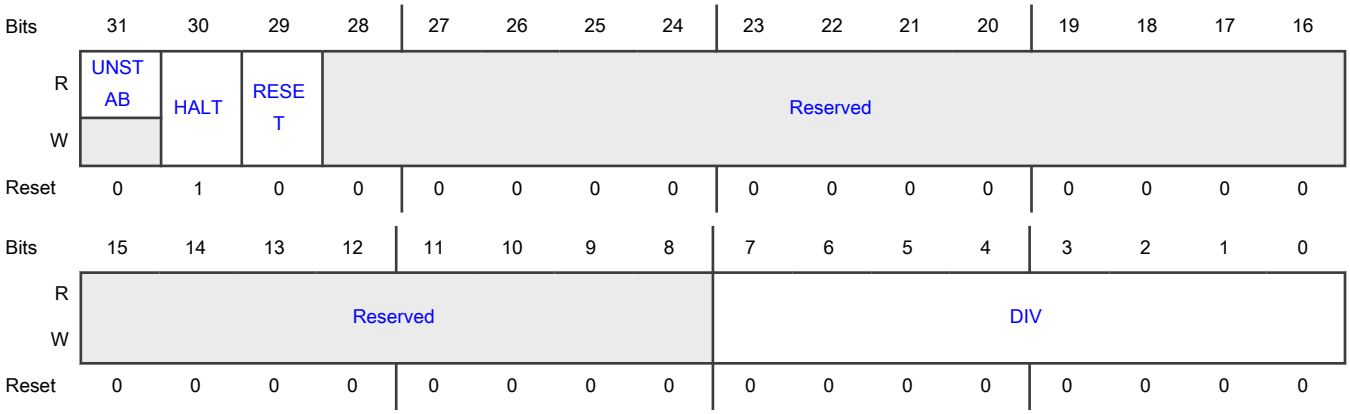
Field	Function
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.113 I3C1 Functional Clock FCLK Divider (I3C1FCLKDIV)

Offset

Register	Offset
I3C1FCLKDIV	B40h

Diagram



Fields

Field	Function
31 UNSTAB	Divider status flag 0b - Divider clock is stable 1b - Clock frequency is not stable
30 HALT	Halts the divider counter 0b - Divider clock is running

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Divider clock is stopped
29 RESET	Resets the divider counter 0b - Divider is not reset 1b - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

17.4.1.114 I3C1 FCLK Slow Selection (I3C1FCLKSSEL)

Offset

Register	Offset
I3C1FCLKSSEL	B44h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														SEL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	I3C1 FCLK Slow Clock Select 000b - FRO_1M clock

Table continues on the next page...

Table continued from the previous page...

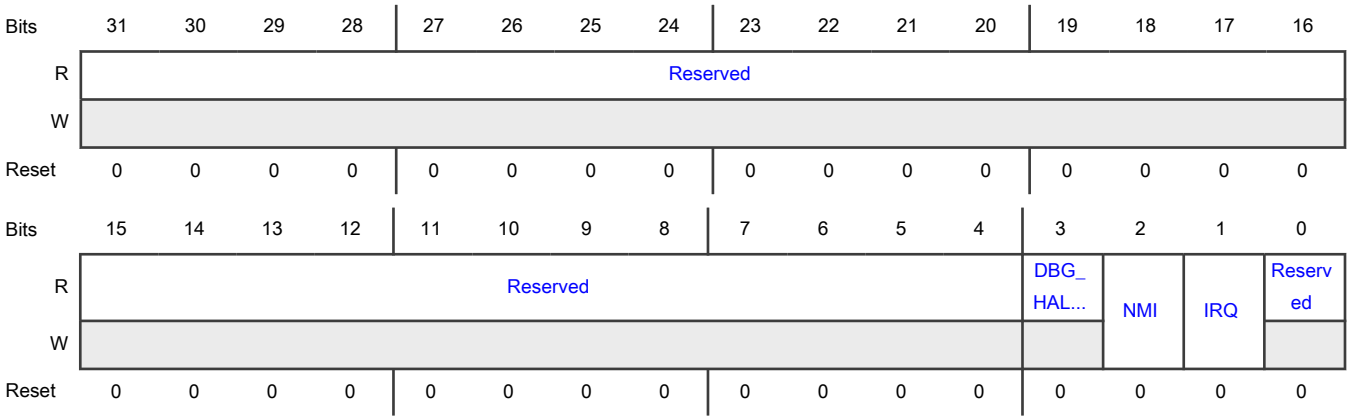
Field	Function
	001b - No clock 010b - No clock 011b - No clock 100b - No clock 101b - No clock 110b - No clock 111b - No clock

17.4.1.115 ETB Counter Status Register (ETB_STATUS)

Offset

Register	Offset
ETB_STATUS	B50h

Diagram



Fields

Field	Function
31-4 —	Reserved
3 DBG_HALT_REQ	Debug halt request 0b - The debug halt request signal is not asserted 1b - The debug halt request signal is asserted when the ETB count expires

Table continues on the next page...

Table continued from the previous page...

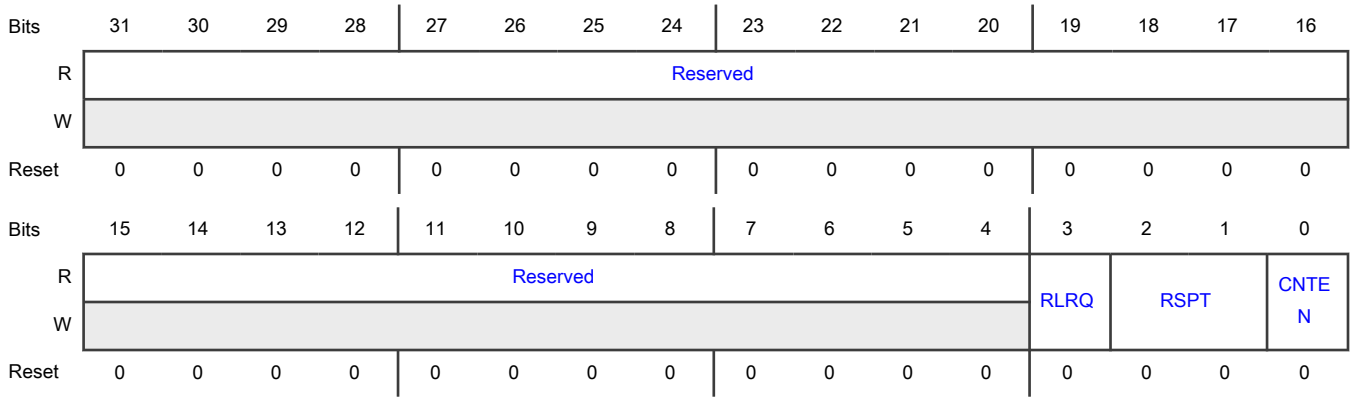
Field	Function
2 NMI	ETB NMI 0b - ETB NMI is not asserted 1b - ETB NMI is asserted. Write 1 to clear it.
1 IRQ	ETB Interrupt 0b - ETB interrupt is not asserted 1b - ETB interrupt is asserted when ETB count expires. Write 1 to clear it.
0 —	Reserved

17.4.1.116 ETB Counter Control Register (ETB_COUNTER_CTRL)

Offset

Register	Offset
ETB_COUNTER_CTRL	B54h

Diagram



Fields

Field	Function
31-4 —	Reserved
3 RLRQ	Reload request Reloads the ETB packet counter with the ETB_COUNTER_RELOAD value. It is always zero when read. If IRQ or NMI interrupts were enabled and an NMI or IRQ interrupt was generated on counter expiration,

Table continues on the next page...

Table continued from the previous page...

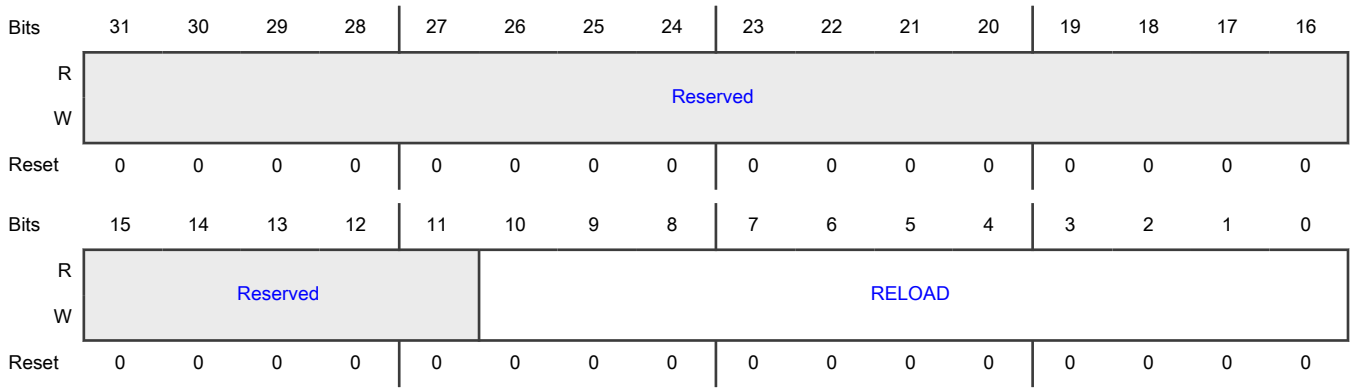
Field	Function
	setting this bit clears the pending NMI or IRQ interrupt request. If debug halt was enabled and a debug halt request was asserted on counter expiration, setting this bit clears the debug halt request. 0b - No effect 1b - Clears pending debug halt, NMI, or IRQ interrupt requests
2-1 RSPT	Response Type 00b - No response when the ETB count expires 01b - Generates a normal interrupt when the ETB count expires 10b - Generates an NMI interrupt when the ETB count expires 11b - Generates a debug halt when the ETB count expires via CPU0 CTICHIN[2]
0 CNTEN	Enables the ETB counter 0b - ETB counter is disabled 1b - ETB counter is enabled

17.4.1.117 ETB Counter Reload Register (ETB_COUNTER_RELOAD)

Offset

Register	Offset
ETB_COUNTER_RELOAD	B58h

Diagram



Fields

Field	Function
31-11	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
10-0 RELOAD	Byte count reload value Indicates the 0-mod-4 value that the counter reloads to. Writing a non-0-mod-4 value to this field results in a bus error.

17.4.1.118 ETB Counter Value Register (ETB_COUNTER_VALUE)

Offset

Register	Offset
ETB_COUNTER_VALUE	B5Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				COUNTER_VALUE											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-11 —	Reserved
10-0 COUNTER_VALUE	Byte count counter value Indicates the current 0-mod-4 value of the counter

17.4.1.119 Gray to Binary Converter Gray code_gray[31:0] (GRAY_CODE_LSB)

Offset

Register	Offset
GRAY_CODE_LSB	B60h

Function

The Gray Code LSB Input register (CODE_GRAY_LSB) contains the least-significant portion of the Gray code to be converted back to binary.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	code_gray_31_0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	code_gray_31_0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 code_gray_31_0	Gray code [31:0] CODE_GRAY_LSB is the least-significant 32 bits of the 42-bit Gray code to be converted.

17.4.1.120 Gray to Binary Converter Gray code_gray[41:32] (GRAY_CODE_MSB)

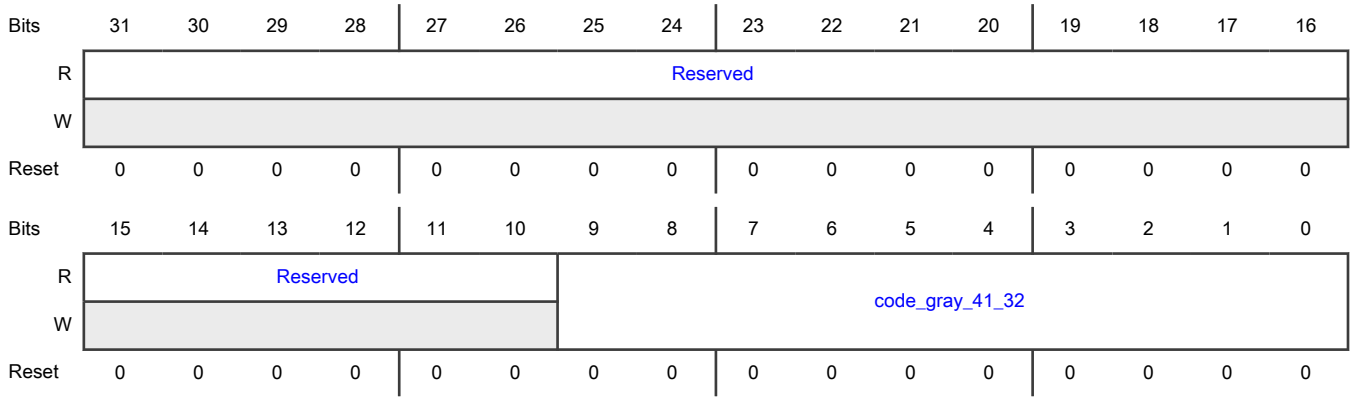
Offset

Register	Offset
GRAY_CODE_MSB	B64h

Function

The Gray Code MSB Input register (CODE_GRAY_MSB) contains the most-significant portion of the Gray code to be converted back to binary.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 code_gray_41_32	Gray code [41:32] CODE_GRAY_MSB is the most-significant 10 bits of the 42-bit Gray code to be converted.

17.4.1.121 Gray to Binary Converter Binary Code [31:0] (BINARY_CODE_LSB)

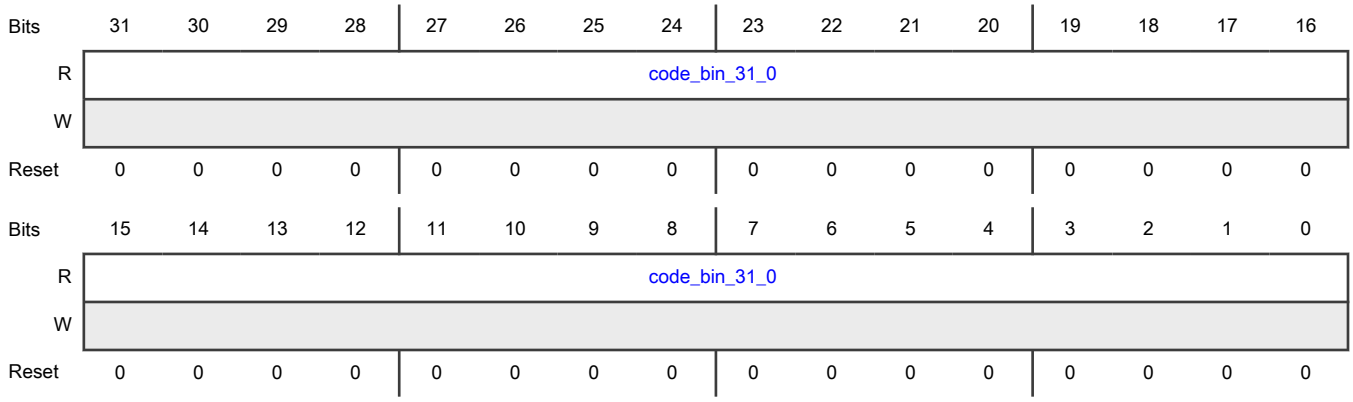
Offset

Register	Offset
BINARY_CODE_LSB	B68h

Function

The Binary Code LSB register (BINARY_CODE_LSB) contains the least-significant portion of the code converted from Gray to binary coding.

Diagram



Fields

Field	Function
31-0 code_bin_31_0	Binary code [31:0] code_bin_31_0 is the least-significant 32 bits of the 42-bit converted code.

17.4.1.122 Gray to Binary Converter Binary Code [41:32] (BINARY_CODE_MSB)

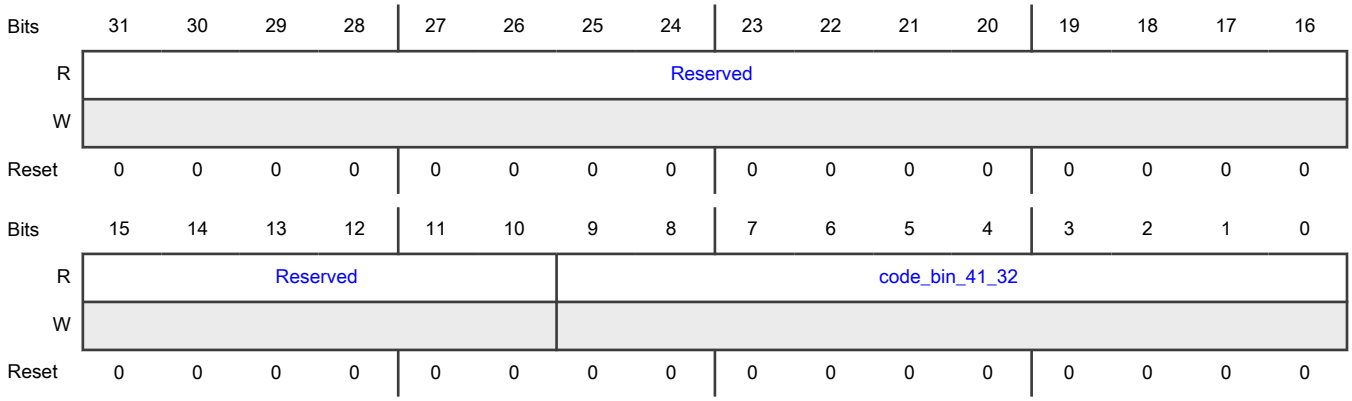
Offset

Register	Offset
BINARY_CODE_MSB	B6Ch

Function

The Binary Code MSB register (BINARY_CODE_MSB) contains the most-significant portion of the code converted from Gray to binary coding.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 code_bin_41_32	Binary code [41:32] code_bin_41_32 is the most-significant 10 bits of the 42-bit converted code.

17.4.1.123 Control Automatic Clock Gating (AUTOCLKGATEOVERRIDE)

Offset

Register	Offset
AUTOCLKGATEOVERRIDE	E04h

Function

This register allows selectively disabling automatic clock gating for device SRAMs. By default, automatic clock gating turns off clocks to each internal SRAM after 16 bus clocks with no activity. This saves power when the SRAMs are not used for a period of time. When turned off due to automatic clock gating, there is a 1 clock delay for the next access to an SRAM. Automatic clock gating may be disabled for time-critical code, which may typically give a 1 or 2% speed improvement.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							RAMH_CT...	RAMG_CT...	RAMF_CT...	RAME_CT...	RAMD_CT...	RAMC_CT...	RAMB_CT...	Reserved	Reserved
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-9 —	Reserved
8 RAMH_CTRL	Controls automatic clock gating for the RAMG Controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
7 RAMG_CTRL	Controls automatic clock gating for the RAMG Controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
6 RAMF_CTRL	Controls automatic clock gating for the RAMF Controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 RAME_CTRL	Controls automatic clock gating for the RAMD Controller. 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
4 RAMD_CTRL	Controls automatic clock gating for the RAMD Controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
3 RAMC_CTRL	Controls automatic clock gating for the RAMC Controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
2 RAMB_CTRL	Controls automatic clock gating for the RAMB Controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
1 —	Reserved Keep the default value.
0 —	Reserved

17.4.1.124 Control Automatic Clock Gating C (AUTOCLKGATEOVERRIDEC)

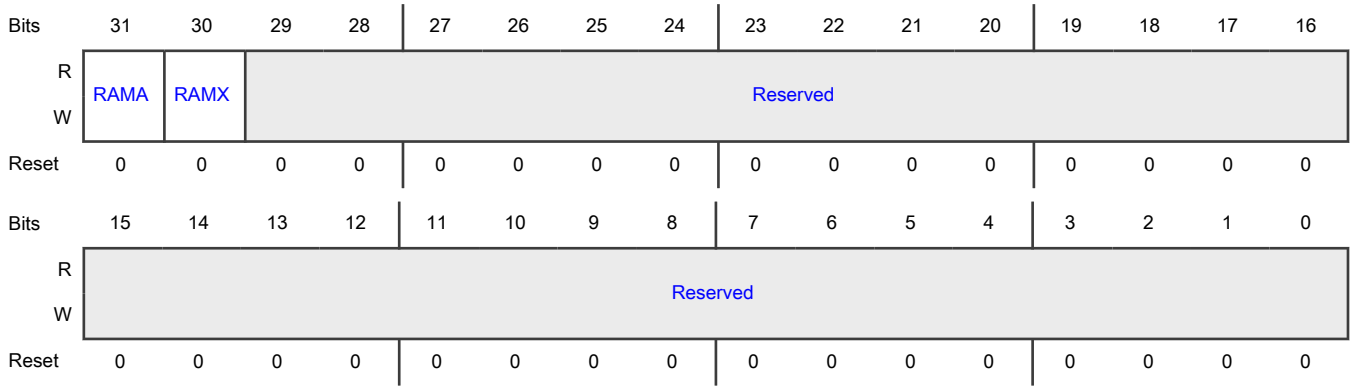
Offset

Register	Offset
AUTOCLKGATEOVERRIDEC	E2Ch

Function

This register allows selectively disabling automatic clock gating for device SRAMs. By default, automatic clock gating turns off clocks to each internal SRAM after 16 bus clocks with no activity. This saves power when the SRAMs are not used for a period of time. When turned off due to automatic clock gating, there is a 1 clock delay for the next access to an SRAM. Automatic clock gating may be disabled for time-critical code, which may typically give a 1 or 2% speed improvement.

Diagram



Fields

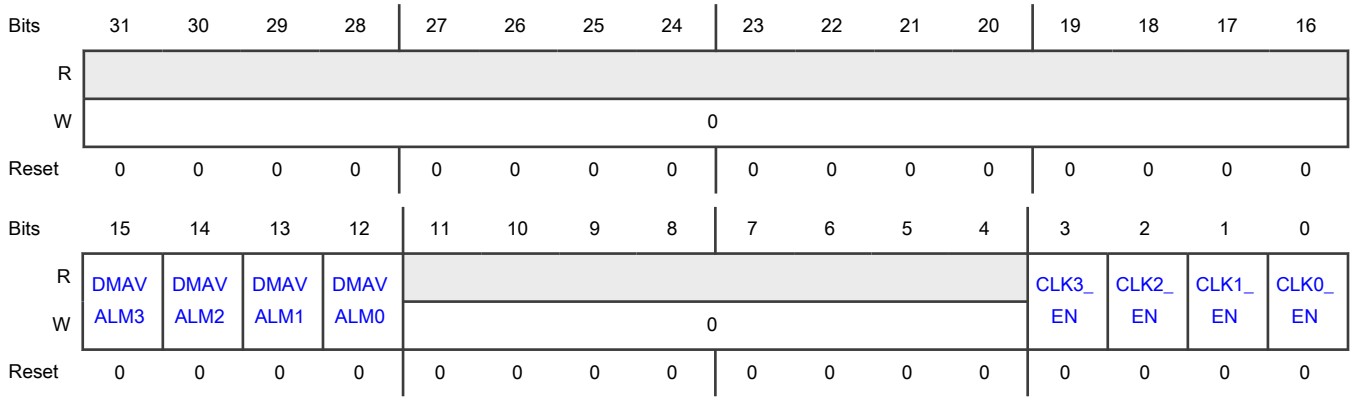
Field	Function
31 RAMA	Controls automatic clock gating of the RAMA controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
30 RAMX	Controls automatic clock gating of the RAMX controller 0b - Automatic clock gating is not overridden 1b - Automatic clock gating is overridden (Automatic clock gating is disabled).
29-0 —	Reserved

17.4.1.125 PWM0 Submodule Control (PWM0SUBCTL)

Offset

Register	Offset
PWM0SUBCTL	E38h

Diagram



Fields

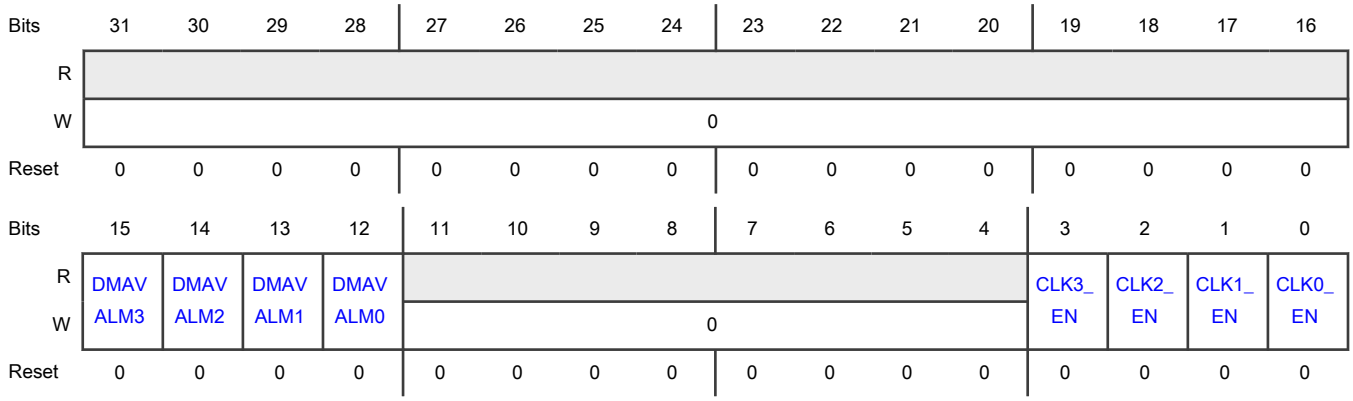
Field	Function
31-16 —	Reserved
15-12 DMAVALMn	PWM0 submodule n DMA compare value done mask When the mask is used, DMA should write the PWM's MCTRL register in the end to set the LDOK bits of relevant submodules at the same time. Since the DMA done signal is blocked, the DMA request is not cleared automatically by the hardware. After the DMA transfer is completed for the current DMA request, the DMA request needs to be cleared manually by clearing SMxDMAEN[VALDE] followed by clearing SMxSTS[RF] with the software.
11-4 —	Reserved
3-0 CLKn_EN	Enables PWM0 SUB Clockn

17.4.1.126 PWM1 Submodule Control (PWM1SUBCTL)

Offset

Register	Offset
PWM1SUBCTL	E3Ch

Diagram



Fields

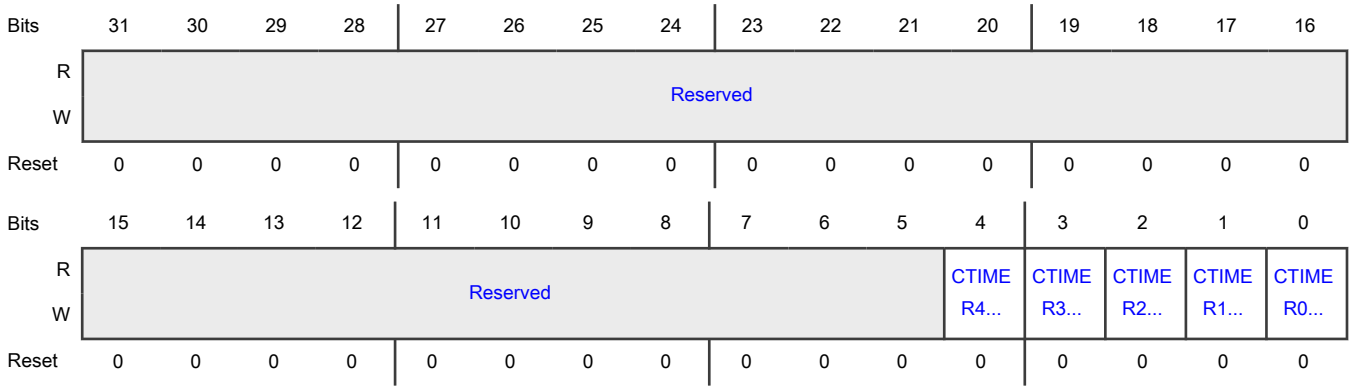
Field	Function
31-16 —	Reserved
15-12 DMAVALMn	PWM1 submodule n DMA compare value done mask When the mask is used, DMA should write the PWM's MCTRL register in the end to set the LDOK bits of relevant submodules at the same time. Since the DMA done signal is blocked, the DMA request is not cleared automatically by the hardware. After the DMA transfer is completed for the current DMA request, the DMA request needs to be cleared manually by clearing SMxDMAEN[VALDE] followed by clearing SMxSTS[RF] with the software.
11-4 —	Reserved
3-0 CLKn_EN	Enables PWM1 SUB Clockn

17.4.1.127 CTIMER Global Start Enable (CTIMERGLOBALSTARTEN)

Offset

Register	Offset
CTIMERGLOBALSTARTEN	E40h

Diagram



Fields

Field	Function
31-5 —	Reserved
4 CTIMER4_CLK_EN	Enables the CTIMER4 function clock 0b - Disable 1b - Enable
3 CTIMER3_CLK_EN	Enables the CTIMER3 function clock 0b - Disable 1b - Enable
2 CTIMER2_CLK_EN	Enables the CTIMER2 function clock 0b - Disable 1b - Enable
1 CTIMER1_CLK_EN	Enables the CTIMER1 function clock 0b - Disable 1b - Enable
0 CTIMER0_CLK_EN	Enables the CTIMER0 function clock 0b - Disable 1b - Enable

17.4.1.128 RAM ECC Enable Control (ECC_ENABLE_CTRL)

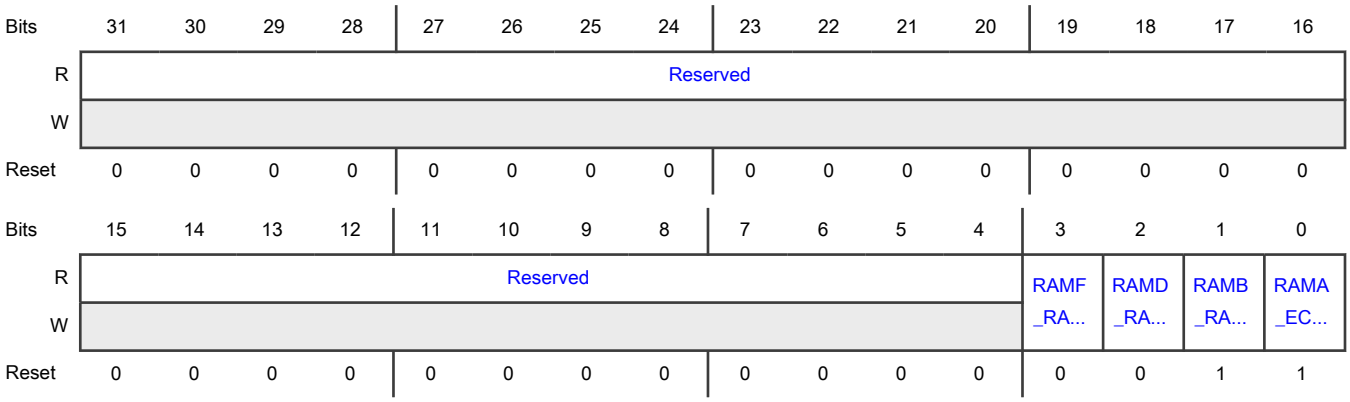
Offset

Register	Offset
ECC_ENABLE_CTRL	E44h

Function

This register is used to enable/disable RAM blocks ECC function. Only combinations 'b0000, 'b0001, 'b0011, 'b0111 and 'b1111 are allowed. For details of RAM blocks and ECC availability for a specific part number refer to the device data sheet .

Diagram



Fields

Field	Function
31-4 —	Reserved
3 RAMF_RAME_ECC_ENABLE	RAMF and RAME ECC enable 0b - ECC is disabled 1b - ECC is enabled
2 RAMD_RAMC_ECC_ENABLE	RAMD and RAMC ECC enable 0b - ECC is disabled 1b - ECC is enabled
1 RAMB_RAMX_ECC_ENABLE	RAMB and RAMX ECC enable 0b - ECC is disabled 1b - ECC is enabled
0 RAMA_ECC_ENABLE	RAMA ECC enable 0b - ECC is disabled 1b - ECC is enabled

17.4.1.129 JTAG Chip ID (JTAG_ID)

Offset

Register	Offset
JTAG_ID	FF0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	JTAG_ID															
W																
Reset	u	u	u	u	0	1	1	1	0	0	1	0	0	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	JTAG_ID															
W																
Reset	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1

Fields

Field	Function
31-0 JTAG_ID	Indicates the device ID

17.4.1.130 Device Type (DEVICE_TYPE)

Offset

Register	Offset
DEVICE_TYPE	FF4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DEVICE_TYPE_PIN								DEVICE_TYPE_PKG				Reserved			DEVICE_...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEVICE_TYPE_NUM															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-24 DEVICE_TYPE_PIN	Indicates the pin number. For example, 100-pin is 0x64.
23-20 DEVICE_TYPE_PKG	Indicates the device package type 0000b - HLQFP 0001b - HTQFP 0010b - BGA 0011b - MAXQFP (HDQFP) 0100b - QFN 0101b - CSP 0110b - LQFP
19-17 —	Reserved
16 DEVICE_TYPE_SEC	Indicates the device secure type 0b - Non-secure part 1b - Secure part
15-0 DEVICE_TYPE_NUM	Indicates the device part number bit[15:12] is family number • 0001b: MCX N series • 0010b: MCX A series • 0011b: MCX L series • other value: reserved bit[11:0]: 3 digital of the part number. For example, the value for the N947 device is 0x947.

17.4.1.131 Device ID (DEVICE_ID0)

Offset

Register	Offset
DEVICE_ID0	FF8h

Function
This register contains the device ID.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				CUSTOMER_REVISION_ID				ROM_REV_MINOR				Reserved			
W																
Reset	0	0	0	0	u	u	u	u	u	u	u	u	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								FLASH_SIZE				RAM_SIZE			
W																
Reset	0	0	0	0	0	0	0	0	u	u	u	u	u	u	u	u

Fields

Field	Function
31-28 —	Reserved
27-24 CUSTOMER_REVISION_ID	CM33_SECURITY_EXTENSION field. 1010b - Non secure version All other values - Any other value represents secure version
23-20 ROM_REV_MINOR	ROM Patch Version.
19-8 —	Reserved
7-4 FLASH_SIZE	Indicates Flash size of the device. 0000b - 32 KB 0001b - 64 KB 0010b - 128 KB 0011b - 256 KB 0100b - 512 KB 0101b - 768 KB 0110b - 1 MB 0111b - 1.5 MB 1000b - 2 MB
3-0 RAM_SIZE	Indicates RAM size of the device. 0000b - 8 KB

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0001b - 16 KB 0010b - 32 KB 0011b - 64 KB 0100b - 96 KB 0101b - 128 KB 0110b - 160 KB 0111b - 192 KB 1000b - 256 KB 1001b - 288 KB 1010b - 352 KB 1011b - 512 KB

17.4.1.132 Chip Revision ID and Number (DIEID)

Offset

Register	Offset
DIEID	FFCh

Function

This register contains the chip number and revision.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				MCO_NUM_IN_DIE_ID											
W																
Reset	u	u	u	u	u	u	u	u	0	1	0	1	0	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MCO_NUM_IN_DIE_ID								MAJOR_REVISION				MINOR_REVISION			
W																
Reset	0	0	0	0	1	0	0	1	1	0	1	0	u	u	u	u

Fields

Field	Function
31-28 —	Reserved
27-8 MCO_NUM_IN_DIE_ID	Chip number
7-4 MAJOR_REVISION	Chip major revision
3-0 MINOR_REVISION	Chip minor revision

Chapter 18
PowerQuad DSP Coprocessor and Accelerator (PowerQuad)

18.1 Chip-specific PowerQuad information

Table 265. Reference links to related information

Topic	Related module	Reference
Full description	PowerQuad	PowerQuad
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

18.1.1 Module instances

This device has one instance of the PowerQuad module, POWERQUAD0.

18.2 Overview

PowerQuad is a DSP Accelerator, designed to assist Cortex-M, working within the Arm ecosystem. The PowerQuad hardware module is designed to accelerate some general DSP computing tasks, such as math, matrix, filter, and transform functions including Fast Fourier Transforms (FFT). Because hardware executes the computing tasks outside the Arm core, tasks run faster and save CPU time.

As technology evolves, so does the need for smarter and lower-power IoT nodes, so the computational performance requirements for local analysis of data is growing. This requirement poses some challenges for the Cortex-M architecture, designed for low power and real-time control. That strength allows a very light layer of software to be able to control I/O and peripherals in a tightly coupled deterministic manner. This same architecture that is advantageous for MCU control, is not optimal for the task of handling vast amounts of data. The purpose of the PowerQuad hardware accelerator is to assist the Cortex-M33 (CM33) core with the handling of digital signal processing.

PowerQuad is a hardware accelerator for common calculations in DSP applications. PowerQuad includes seven internal computation engines:

- Transform
- Transcendental function
- Trigonometry function
- Dual Biquad infinite impulse response (IIR) filter
- Matrix accelerator
- Finite impulse response (FIR) filter
- Coordinate rotation digital computer (CORDIC)

By significantly reducing the runtime and the energy requirements of signal processing algorithms, PowerQuad allows you to run more accurate, elaborate algorithms in an energy-efficient manner.

18.2.1 Block diagram

PowerQuad is a digital signal coprocessing companion to the CM33 core. You can access PowerQuad through the coprocessor interface, the memory mapped interface, or through the CORDIC interface address space.

PowerQuad is primarily based on a quad floating-point multiply accumulator (MAC) compute block. This four-way MAC implements almost all functions supported by the accelerator. The other compute elements in PowerQuad are the FFT radix-8 butterfly engine (for FFT/DCT calculations) and the CORDIC for inverse trigonometric functions (arctan, arctanh).

Since the main compute engine is natively in floating-point format, this simplifies operations that are not tightly bound in terms of input data or result. Having a floating-point format ensures that a certain resolution can be achieved without the need for saturating the checking and/or implementing logic.

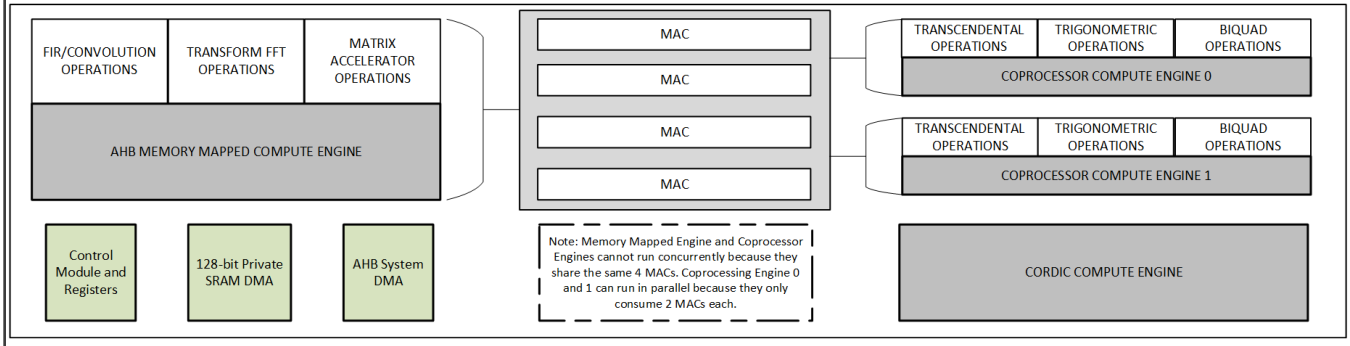


Figure 92. PowerQuad block diagram

18.2.2 Features

The key features of PowerQuad are:

- Communication interfaces:
  - Coprocessor interface from the CM33 core, a register-to-register interaction.
  - AHB as a subordinate interface, use to configure the PowerQuad registers.
  - AHB as a manager interface for PowerQuad to fetch or write data to and from the system memory, when required.
  - 128-bit wide RAM controller that can access up-to 4 RAM banks simultaneously to fetch or write data as needed.
- Four single-precision floating-point multiply accumulator (MAC) units
- Coprocessor engines for tightly coupled opcodes: sin(x), cos(x), ln(x), e^(x), e^(-x), 1/(x), 1/sqrt(x), sqrt(x), biquad(x). Using two MACs, you can run two opcodes in parallel
- Memory-mapped
  - FFT/iFFT/DCT/iDCT operations
  - Matrix operations: add, sub, dot, prod, mult, inverse, transpose, scale
  - Convolution/Correlation/FIR
- Arctan/Arctanh (these operations can be used as the base to compute any CORDIC function)

With the assistance of PowerQuad, the CM33 core is free to perform other tasks. While PowerQuad executes an assigned computation task, the CM33 core can prepare the next PowerQuad task, resulting in a pipeline of PowerQuad tasks.

18.3 Functional description

In a conventional MCU without PowerQuad, the main CPU performs all of the computation work. When using the PowerQuad, work is accelerated and depending on the operation in progress the main CPU can continue to work on other tasks while the PowerQuad finishes processing the operation. For example, transcendental functions are now essentially assembly instructions

for the CM33 core. Other larger functions (such as FFT, or matrix calculations) can be considered as a kind of DMA job, where the core can start them and receive an interrupt or event when the job completes.

Using PowerQuad, you can treat PowerQuad as a coprocessor, performing multiple tasks by rearranging them and reducing the branching to speed up computational processing. The PowerQuad engines compute the assigned functions faster than using the CM33 core alone.

The aim of the PowerQuad coprocessor is to serve the following purposes:

- Provide significant energy reduction for the execution of digital signal processing algorithms
- Significantly accelerate digital signal processing operations compared to the Cortex-M core alone
- Provide hardware acceleration for common DSP functions (mainly from the CMSIS-DSP library)

After applying the clock to PowerQuad, based on the requested operation, one of the computation engines activates. All PowerQuad operations fall into one of three access types. Access PowerQuad through either:

- The coprocessor engines(CP)
- The AHB memory mapped engine (AHB).
- The CORDIC engine

18.3.1 Coprocessor interface

18.3.1.1 Introduction

The ARMv8-M architecture (CM33) introduces a coprocessor interface, which allows PowerQuad access through opcodes:

- MCR (Move from coprocessor to Register)
- MRC (Move from Register to coprocessor)

When accessing PowerQuad through the coprocessor interface, executing a correctly formatted coprocessor instruction launches the operation. Executing a second coprocessor instruction retrieves the result. The coprocessor is self-stalling, which means that if you perform a read from a result register, the access stalls until the computation is complete. These take 2 MACS out of the 4 to implement, allowing you to run two operations in parallel.

To achieve the best performance, use vectorized forms of the coprocessor instructions. These typically handle the work at function calls and follow the timing as shown in the figure below. These vectorized forms are already coded by NXP to achieve timing as shown (note that the figure shows just 2 calculations. In the vectorized functions, we perform 4 of these in a row (that is, 8 calculations).

When submitting the data and/or opcodes to a coprocessor, the CM33 core can continue executing other tasks while the coprocessor computes in parallel. This parallel work ensures that the CM33 core and PowerQuad are not left idle, maximizing the performance of the work done.

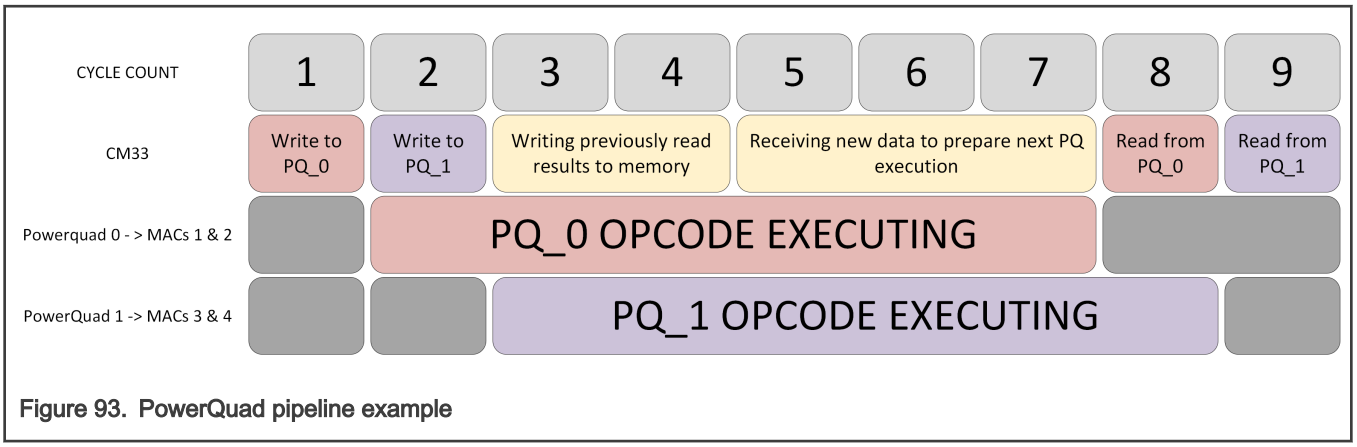


Figure 93. PowerQuad pipeline example

In PowerQuad, this interface uses short (several cycles) operations, for which the CM33 core provides data from one of its registers. The result is stored back in the CM33 register. The above diagram represents a snippet in time where the CM33 core and PowerQuad work in parallel. The CM33 core requests that PowerQuad 0 begin to execute an opcode provided. In the following cycle the CM33 core requests that PowerQuad 1 also begin to execute an opcode. Meanwhile, PowerQuad is fully loaded and the CM33 core takes this time to write back any data previously received from PowerQuad. The CM33 core also uses this time to prepare the next PowerQuad requests. Once PowerQuad 0 finishes, the CM33 core fetches the data by reading back the result in one cycle. Then in the next cycle reads back the result from PowerQuad 1. You can repeat this pipelined effort continuously to optimize the workload on the CM33 core while PowerQuad is busy.

When the data comes to PowerQuad in fixed-point or floating-point format, internally PowerQuad handles the data as floating point. The conversion from fixed-point to floating-point format occurs automatically. Prescaling and postscaling options are available to allow fixed-point numbers with fractional precision.

Results of PowerQuad operations can be either fixed-point or floating-point format (conversion from floating-point to fixed-point format, if needed).

The functions handled in this way are:

- Transcendental: ln(x), e^(x), e^(-x), 1/(x), 1/sqrt(x), sqrt(x), x1/x2
- Trigonometric: sin(x), cos(x)
- Biquad: biquad(x)

For PowerQuad Coprocessing interface examples, see Construct coprocessor interface operations.

18.3.1.2 Transcendental

In mathematics, a transcendental function is a type of function that you cannot express algebraically. A possible solution to simplify the implementation of these algorithms is to create a lookup table. Lookup tables can help to simplify or speed up an inefficient process, however lookup tables take extra space in the memory of the device. A key advantage of PowerQuad is that there is no need for lookup tables, so you do not waste RAM for transcendental functions. All of this work is taken care of by the hardware module.

Another advantage of transcendental function implementation for PowerQuad is that they become actual instructions in the CM33 core. The functions give high precision in a matter of a handful of cycles.

Since PowerQuad can execute 2 of these functions in parallel, the number of cycles consumed matches the number of cycles the CM33 core requires to bring in more data for PowerQuad and write out the results. Therefore, you can fully pipeline PowerQuad running in parallel with the CM33 core and achieve almost single-cycle effective performance for the calculations.

PowerQuad supports the following transcendental functions:

- ln(x)
- $e^{(x)}$
- $e^{(-x)}$
- 1/(x)
- 1/sqrt(x
- sqrt(x)
- x1/x2

18.3.1.2.1 Requirements

Input and output data support float, INT16, and INT32.

18.3.1.2.2 Performance

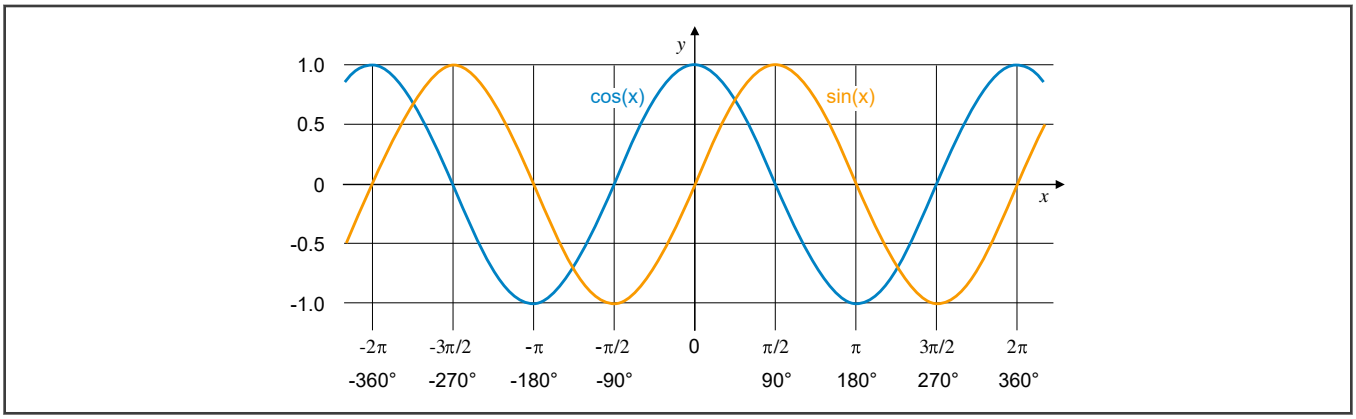
Table 266. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Transcendental	PQ_LnF32	15.0E-6	80.0E-9	187.6	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed Note, the math functions running on CM33 are only supported on math.h
	PQ_SqrtF32	4.2E-6	93.3E-9	45.3	
	PQ_InvSqrtF32	4.2E-6	80.0E-9	52.8	
	PQ_EtoxF32	18.8E-6	86.7E-9	217.2	
	PQ_EtonxF32	18.8E-6	86.7E-9	217.2	
	PQ_LnFixed	15.0E-6	60.0E-9	250.2	
	PQ_SqrtFixed	4.2E-6	73.3E-9	56.9	
	PQ_InvSqrtFixed	4.2E-6	66.7E-9	62.7	
	PQ_EtoxFixed	19.4E-6	60.0E-9	324.0	
	PQ_EtonxFixed	19.4E-6	73.3E-9	264.3	

18.3.1.3 Trigonometry

The trigonometric functions most widely used in modern mathematics are the sine, the cosine, and the tangent. In embedded applications, trigonometric functions are used in robotics, motor control, signal processing among others. It is essential to process data quickly to benefit from the timing aspects of each application.

The Taylor series of sin(x) and cos(x) centered about 0 have the property of having best convergence (fewest terms) when closest to 0. By partitioning the angular space into half-quadrants (that is π/2 or 45 degrees), you can reuse the basic Taylor series for the sin(x) and cos(x) in the range [0, π/2].



Tangent can be calculated using the equation:

$$\tan(x) = \frac{\sin(x)}{\cos(x)} = \frac{\sin(x)}{\sqrt{1 - sin^2(x)}}$$

The tangent function is not directly mapped as a PowerQuad function. To calculate the tangent of a given value, you can perform a sine and cosine function in parallel and then perform the division of those results. In addition, you could also do sine and inverse square root, however, this method would consume more cycles than the first option.

18.3.1.3.1 Requirements

Input and output floating-point data support radian units, Q15 and Q31 fixed-point data support normalized values.

For fixed-point inputs, the input value is normalized to range -1 to 1 which represents -pi to pi.

For floating-point inputs, the input value is in radians. This function calculates -pi to pi.

18.3.1.3.2 Performance

Table 267. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Trigonometry	PQ_SinF32	686.7E-9	206.7E-9	3.3	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	PQ_CosF32	720.0E-9	180.0E-9	4.0	
	PQ_SinQ15	413.3E-9	286.7E-9	1.4	
	PQ_CosQ15	453.3E-9	233.3E-9	1.9	
	PQ_SinQ31	493.3E-9	233.3E-9	2.1	
	PQ_CosQ31	480.0E-9	226.7E-9	2.1	

18.3.1.4 Dual Biquad IIR

In signal processing, a digital Biquad filter is a second order recursive linear filter, containing two poles and two zeros.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

The coefficients are often normalized such that $a_0 = 1$:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

High-order IIR filters can become unstable due to quantization of the coefficients. Typically higher-order filters are implemented in cascaded Biquad portions. PowerQuad can execute cascaded filters by saving the internal states of each run and restoring the states as the data is processed.

The Biquad engine uses the coprocessor interface. Use the APIs provided in the SDK, as executing a Biquad filter is composed of several steps. The MCR and MRC opcodes are used sequentially depending on the order of the filter used. A second order recursive digital filter is shown below.

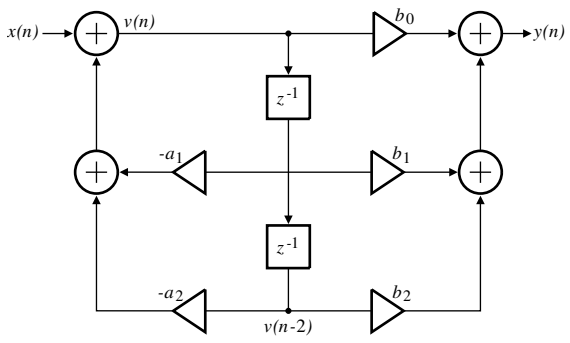


Figure 94. Second-order IIR filter

18.3.1.4.1 Requirements

Input and output data support float, INT16, and INT32.

For single Biquad computations, Biquad state 0 is used.

For cascade Biquad computations, the data flow is input -> Biquad state 1 -> Biquad state 0 -> output.

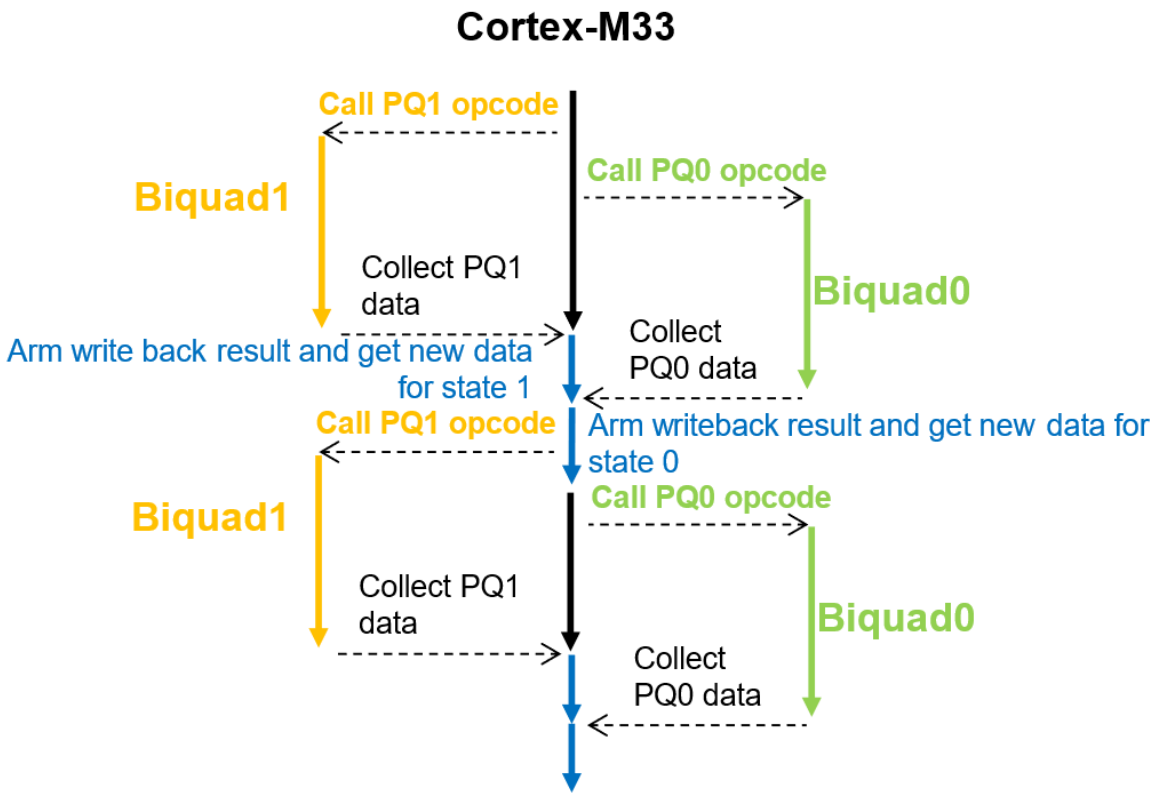


Figure 95. Biquad pipeline example

18.3.1.4.2 Performance

Table 268. Performance using PowerQuad

Function	API	Using PowerQuad (s)	Conditions
Biquad IIR	PQ_BiquadF32	5.6E-6	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed SW Math Comparison does not exist at the moment.
	PQ_BiquadIncrementalF32	4.2E-6	
	PQ_BiquadFixed16	1.5E-6	
	PQ_BiquadFixed32	1.3E-6	
	BiquadCascadeFixed16	4.0E-6	
	BiquadCascadeFixed32	3.9E-6	
	BiquadCascadeF32	89.7E-6	

18.3.2 AHB memory-mapped engine

18.3.2.1 Introduction

PowerQuad handles large data operations in a memory-mapped fashion. To access PowerQuad through this method, write to Control (CONTROL) to launch a memory-mapped operation. When the result is ready, an event or interrupt generates to inform the CM33 core that the result is ready. Alternatively, the CM33 core can also poll CONTROL[INST_BUSY]. Retrieve results through one or more AHB reads from the PowerQuad's private RAM or the system's memory.

Some operations must use system memory for intermediate operations or to save context while using the PowerQuad for another operation. Handle these operations through the 32-bit AHB bus. Most memory-mapped operations use the private 128-bit wide memory. Use both the AHB bus and private memory if possible. That is, have one operand coming from system memory and one from private memory, such that you maximize bandwidth and performance. Move operands into private memory by choosing private memory as the result destination pointer. The private memory always keeps the data in the native format (which is floating point in all cases except for FFT, which uses fixed point). In FFT, we recommend that you use private memory for the intermediate calculations, as a temporary region. For other calculations, the recommendation is that one side comes from AHB (system) and the other from private memory. In order to get the operand into private memory, use a matrix scale operation (scale by 1.0) and have an address in private memory as the destination.

PowerQuad is natively 32-bit floating-point, so temporary results in private memory can only be in the same format. The only exception is the FFT engine, which uses fixed-point data. The 4-region memory model configures these parameters:

- Scaling and data format for the operands
- Temporary results
- Final result to fixed-point or floating-point format

With PowerQuad, you can change the format, prescale, or postscale the data on the fly, which is a huge advantage, saving cycles to increase performance.

The CM33 core writes to the control registers of PowerQuad to:

- Select the function
- Provide 4 memory regions with source destination addresses
- Save a temporary scratch address for intermediate results (where applicable such as with FFT and Matrix Inversion)

Poll completion in CONTROL[INST_BUSY] (cleared to ‘0’), or generate an interrupt/event from the PowerQuad.

The functions handled in this way are:

- Matrix (add, sub, scale, invert, transpose, multiply, dot, product)
- FFT/IFFT/DCT/IDCT (note that DCT is partial in PowerQuad and requires some adjustments)
- FIR/Convolution/Correlation

For PowerQuad AHB memory-mapped interface examples, see Construct AHB interface operations .

For any given function where PowerQuad interacts with data (from AHB system memory or private RAM), a user set configuration for 4 regions gives the addressing and the interface for interacting with the data.

The regions are:

- Output
- Temp
- Input A
- Input B

The PowerQuad control registers allow the user to choose the base address for each of the regions used in the function. Apart from a base address for each region, the user can also choose the format for the data on the inside and the outside. For example, you can configure Input A as INT16 fixed-point input but use it as floating point by PowerQuad. The format in or out can only be chosen if the region used is in AHB space. All data written to or read from private RAM does not undergo any conversion.

The internal or external formats can be any of the following:

- Int16 Fixed point
- Int32 Fixed point
- Floating point

In addition to formatting the input and output, you can also choose a scaling factor for the fixed-point inputs and outputs.

18.3.2.2 Memory Handling

PowerQuad has four memory pointer regions:

Input A pointer	Stores the argument A for the algorithm
Input B pointer	Stores the argument B for algorithm (optional)
Temp pointer	Area used for intermediate calculations
Output pointer	Where the result is stored

Each of the memory regions may be configured in the following manner:

Origin data	32-bit fixed, 16-bit fixed, or 32-bit float
Data used in PowerQuad	All data must be float except in the FFT engine which are fixed.
Scaling	PowerQuad can do scaling by power of 2 for input or output.

Each of these regions may reside in internal memory or may reside in private memory. PowerQuad accesses and collects the data accordingly. It is able to access AHB memory space for the entire address space, except for the range E000_0000h – EFFF_FFFFh. The address space starting with E000_0000h is that through which, PowerQuad can access its private RAM. It is recommended to make use of private memory as this region is only accessible by PowerQuad.

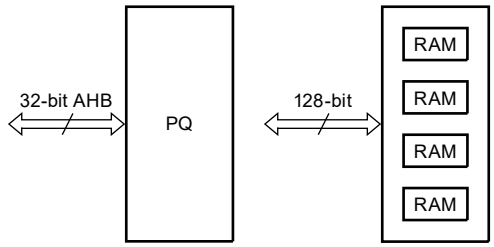


Figure 96. PowerQuad using private RAM as the 128-bit wide scratch pad

18.3.2.3 Transform

PowerQuad provides discrete Fourier transforms and discrete cosine transforms, implemented with a Radix-8 Butterfly structure Fast Fourier Transform (FFT) engine using fixed-point arithmetic at a resolution of 24 bits. The Radix-8 butterfly structure of the engine is shown below. This implementation both reduces the number of memory accesses and makes full use of the four MACs available in PowerQuad.

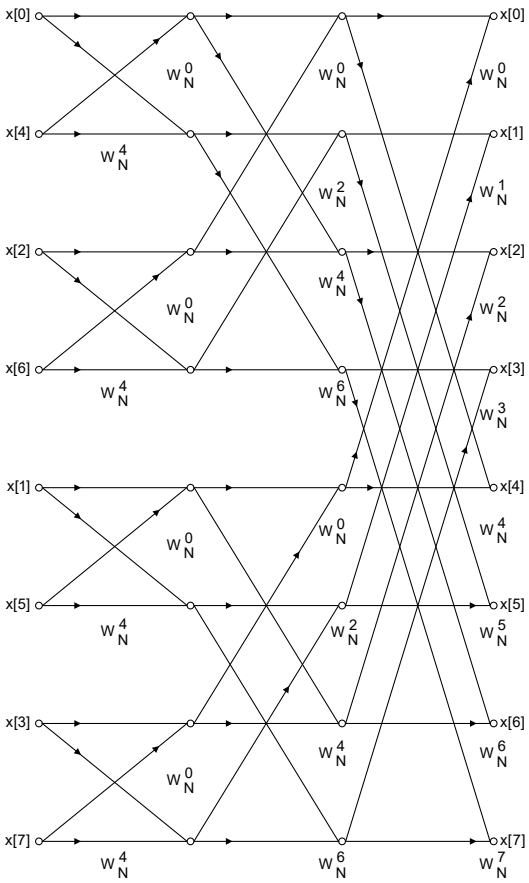


Figure 97. Radix-8 butterfly structure

The discrete Fourier transform (DFT) transforms a sequence of N complex numbers:

$$\{x_n\} := x_0, x_1, \ldots, x_{N-1}$$

Into another sequence of complex numbers:

$$\{X_k\} := X_0, X_1, \ldots, X_{N-1}$$

Defined by:

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n \bullet e^{-i2\pi kn/N}$$

With the inverse transform defined as:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \bullet e^{i2\pi kn/N}$$

The discrete cosine transform is similar to the discrete Fourier transform but uses only real numbers. It transforms a sequence of N real numbers:

$$\{x_n\} := x_0, x_1, \ldots, x_{N-1}$$

into another sequence of real numbers:

$$\{X_k\} := X_0, X_1, \ldots, X_{N-1}$$

Defined by:

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], where\ k = 0, \ldots, N-1$$

And the inverse defined by:

$$X_k = \frac{1}{2}x0 + \sum_{n=1}^{N-1} x_n cos\left[\frac{\pi}{N}n\left(k+\frac{1}{2}\right)\right], where\ k = 0, \ldots, N-1$$

FFT and IFFT are fully performed by PowerQuad. In the case of DCT, PowerQuad only performs odd/even mirroring of the original input before performing an FFT on the input. For example, input of [0 1 2 3 4 5 6 7] gets odd/even mirroring that results in [0 2 4 6 7 5 3 1] before entering FFT. To reach a final DCT value, ARM is required to multiply the resulting spectrum with a rotating phasor. For IDCT, ARM needs to multiply the input by a phasor. The PowerQuad IDCT machine then performs an inverse FFT and then the reverse of the odd/even mirroring done by the transform.

18.3.2.3.1 Requirements

Input and output data support INT16 and INT32.

Important aspects of the input data for the FFT engine are:

- Only considers the bottom 27 bits of the input word, therefore prescaling must not exceed this limit to avoid saturation.
- Always scales the input data by 1/N when computing. If an unscaled result is necessary, the input data (in the INPUT A region) must first be multiplied by N. The inverse FFT is also scaled by 1/N, but is correct per the formula, so no scaling treatment is needed.
- Achieves scaling (that is, multiplying by N) at no cost by configuring the prescaling for the input into FFT.
- Supports these lengths for DCT and DFT are N = 16, 32, 64, 128, 256, 512.

NOTE

PowerQuad can perform scaling by N on the way in with no impact to performance.

Real number sequences must be arranged in memory such that if a sequence is as follows:

$$\{x_n\} := x_0, x_1, \ldots, x_{N-1}$$

Then the input array in memory is:

$$x[N] = \{x_0, x_1, \ldots, x_{N-1}\}$$

Complex number sequences must be arranged in memory such that if a sequence is as follows:

$$\{x_n\} := x_0, x_1, \ldots, x_{N-1}, where\ x_n = x_{nreal} + x_{nim}$$

Then the input array in memory is:

$$x[N] = \{x_0 real, x_0 im, x_1 real, x_1 im, \ldots, x_{N-1} real, x_{N-1} im\}$$

For the output sequence, the array is always made up of complex numbers, where the imaginary parts are zero for real number operations.

18.3.2.3.2 Performance

Table 269. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
FFT	rfft_q15	446.6E-6	20.1E-6	22.2	Frequency 150 MHz

Table continues on the next page...

Table 269. Performance using CM33 alone versus with PowerQuad (continued)

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
	rfft_q31	537.5E-6	20.1E-6	26.7	IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	cfft_q15	674.9E-6	22.2E-6	30.4	
	cfft_q31	830.4E-6	22.2E-6	37.4	
	ifft_q15	675.5E-6	22.1E-6	30.5	
	ifft_q31	854.4E-6	22.1E-6	38.6	

18.3.2.4 Matrix

The matrix accelerator engine supports the following operations:

- Addition
- Subtraction
- Multiplication
- Vector Dot-Product
- Inversion
- Transpose
- Scaling

Matrices are composed in the following manner:

In the example matrices below, Matrix A, is composed of 2 rows and 3 columns. Matrix B is composed of 2 rows and 2 columns. Depending on the required matrix operation, the number of rows and columns of each matrix determines if it is possible to proceed.

$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

Adds/Subtracts/product of two matrices:

The number of rows in Matrix A must match the rows in Matrix B. The same applies to the columns of each matrix.

Multiplication of two matrices:

The number of columns in Matrix A must match the number of rows in Matrix B.

Scaling of a matrix:

A matrix is scaled using a floating-point scaler number.

[R] = k * [M], where each element in the matrix is scaled by k. The floating-point scaler number is loaded into the Miscellaneous (MISC) register.

Transpose of a matrix:

The transpose of a matrix is an operator that flips a matrix over its diagonal, that is, it switches the row and column indexes of the matrix by producing another matrix denoted as A^T.

Let A be an m × n matrix. The transpose of A, denoted A^T, is the n × m matrix whose columns are the respective rows of A.

Inversion of a matrix:

Matrix inversion is supported only for square matrices. A square matrix is singular only when its determinant is exactly zero. The implementation needs some scratch area in the AHB space. We are not using any of private RAM for this function.

Vector Dot Product:

The dot product is the sum of the products of the corresponding entries of the two sequences of numbers (vectors). The result is one number.

18.3.2.4.1 Requirements

Input and output data support float, INT16, and INT32.

The Length (LENGTH) register is remapped when using the matrix engine. It is composed of the concatenation of the columns in matrix_B, the columns in matrix_A, and the rows in matrix_A.

- LENGTH[31:0] = Cols_B[20:16] | Cols_A[12:8] | Rows_A[4:0]

Matrix operations have a maximum dimension as shown below:

Table 270. Matrix operation dimensions

Operation	Maximum size
Addition	16 x 16
Subtraction	
Hadamard Product	
Product	
Transpose	
Scaling	
Inversion	9 x 9
Vector Dot-Product	256 elements

18.3.2.4.2 Performance

Table 271. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Matrix	mat_add_q15	12.2E-6	6.2E-6	2.0	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	mat_add_q31	12.2E-6	6.2E-6	2.0	
	mat_add_f32	19.4E-6	6.2E-6	3.1	
	mat_sub_q15	12.2E-6	6.2E-6	2.0	
	mat_sub_q31	12.2E-6	6.1E-6	2.0	
	mat_sub_f32	19.4E-6	6.2E-6	3.2	
	mat_mult_q15	322.4E-6	19.3E-6	16.7	
	mat_mult_q31	346.9E-6	19.4E-6	17.9	
	mat_mult_f32	352.5E-6	43.7E-6	8.1	

Table continues on the next page...

Table 271. Performance using CM33 alone versus with PowerQuad (continued)

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
	mat_inverse_f32	118.4E-6	34.9E-6	3.4	
	mat_trans_q15	10.4E-6	5.1E-6	2.1	
	mat_trans_q31	10.4E-6	5.1E-6	2.0	
	mat_trans_f32	16.8E-6	6.2E-6	2.7	
	mat_scale_q15	17.3E-6	5.1E-6	3.4	
	mat_scale_q31	33.6E-6	5.2E-6	6.5	
	mat_scale_f32	15.6E-6	4.4E-6	3.6	

18.3.2.5 FIR

In digital signal processing, an Nth order discrete time FIR filter does not have feedback, which makes it a very stable filter. However, this type of filter requires many more coefficients to meet a similar outcome of an IIR filter. Due to this requirement, the time it takes to execute a FIR filter is much longer as each coefficient introduces another multiplication in the system. Having PowerQuad as an accelerator of the same operation greatly reduces the impact of having a high-order FIR filter.

The signal flow graph for the general, causal, length N = M + 1, finite-impulse-response (FIR) digital filter is shown below.

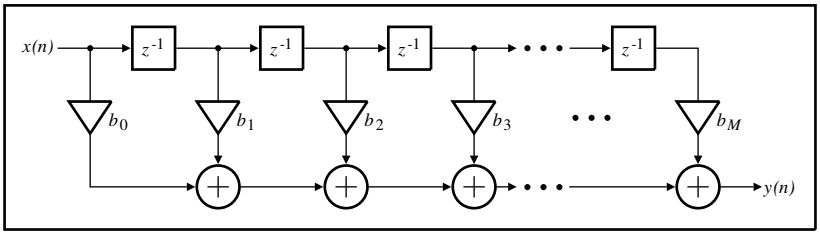


Figure 98. FIR signal flow

This filter implements the difference equation:

$$y[n] = \sum_{i=0}^{M} b_i x[n-i], \text{ where } n = 0, \ldots, M$$

Here x[n] is the input signal, y[n] is the output signal, and M is the filter order. An Mth-order filter has (M + 1) filter taps.

The PowerQuad FIR engine run 4 MAC operations in parallel.

18.3.2.5.1 Requirements

Input and output data support float, INT16, and INT32.

Length (LENGTH) is remapped when using the FIR engine. It is composed of the concatenation of the number of filter taps and the number of samples in the input sequence.

- LENGTH[31:0] = numTaps[31:16] | numSamples[15:0]

Configuring the FIR incremental operations uses the offset from the base address that the FIR operation uses. FIR incremental operation may only be used after FIR operation is used. Prescaling and postscaling the data can be a huge advantage for these operations.

- PQ_FIR(POWERQUAD, pSrcA, srcALen, pSrcB, srcBLen, pDst, op_type);
- PQ_FIRIncremental(POWERQUAD, srcALen, srcBLen, offset);

For full API examples, check the SDK PowerQuad drivers provided at mcuxpresso.nxp.com.

See the MCUXpresso SDK API Reference Manual included in the SDK document folder SDK.zip/docs.

18.3.2.5.2 Performance

Table 272. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Filter	fir_q15	20.5E-6	1.9E-6	10.6	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	fir_q31	9.9E-6	1.9E-6	5.1	
	fir_f32	16.7E-6	1.8E-6	9.1	
	conv_q15	4.0E-6	1.4E-6	2.8	
	conv_q31	3.9E-6	1.4E-6	2.7	
	conv_f32	4.3E-6	1.4E-6	3.1	
	correl_q15	4.8E-6	1.5E-6	3.2	
	correl_q31	4.0E-6	1.5E-6	2.7	
	correl_f32	4.2E-6	1.4E-6	3.0	

18.3.3 CORDIC engine

CORDIC, also known as Volder's algorithm, is a simple and efficient algorithm to calculate many different functions including trigonometric, hyperbolic, and logarithmic functions; and real and complex multiplications.

CORDIC is helpful in accelerating computations in various applications such as graphics in human-machine interfaces and field-oriented control. PowerQuad adds to the advantages of using CORDIC by accelerating the iterations per cycle.

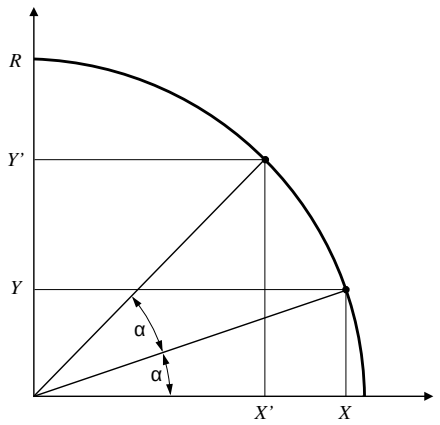


Figure 99. Rotation and vectoring mode of the CORDIC algorithm

In PowerQuad, the CORDIC engine can be directly used for inverse tangent and inverse hyperbolic tangent functions.

CORDIC hardware implements 2 iterations per cycle. The number of iterations is programmable. Although the iterations may be programmable, there is a tradeoff of speed vs. accuracy. CORDIC also allows use cases like calculating phase and magnitude from a real/imaginary pair simultaneously with 2 bits of convergence of output accuracy per cycle. In addition, the CORDIC engine stalls the CPU until the result is ready. Therefore, it is not necessary to poll for events or interrupts.

Functions that can use CORDIC:

- Arctan (x) : 12 cycles for 24-bit accuracy
- Arctanh (x) : 12 cycles for 24-bit accuracy

For PowerQuad CORDIC interface examples, see CORDIC API examples.

18.3.3.1 Requirements

Input and output data support INT16 and INT32.

18.3.3.2 Performance

Table 273. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
CORDIC	PQ_ArctanFixed	19.3E-6	486.7E-9	39.7	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed Note, the math functions running on CM33 are only supported on math.h
	PQ_ArctanhFixed	28.1E-6	453.3E-9	62.0	

18.3.4 Clocking

Enable the PowerQuad clock through an AHB Clock Control register in the system control of your chip.

See PowerQuad chip-specific clocking.

18.3.5 Interrupts

Events and Interrupts are enabled in the PowerQuad registers for AHB operations. PowerQuad informs the Cortex-M33 core, based on event or interrupt, upon completion of the chosen algorithm. Events and interrupts are enabled in the PowerQuad registers for AHB interface operations, they cannot be enabled for coprocessor interface operations. PowerQuad informs the CM33 core through the AHB bus, based on an event or interrupt, upon completion of a chosen algorithm.

18.4 External signals

This module has no external signals.

18.5 Initialization

See PowerQuad chip-specific initialization.

1. Enable the PowerQuad clock in the system configuration of your chip.
2. Reset the PowerQuad peripheral in the system configuration of your chip.
3. Enable events and interrupts.

18.6 Application information

This section explores relevant examples for both the coprocessor and AHB interface operations.

18.6.1 Construct coprocessor interface operations

MCR and MRC opcodes are constructed below for each of the supported algorithms in the coprocessor mechanism. The APIs in the SDK drivers internally use the MCR and MRC opcodes. Use these APIs, which are provided in the SDK drivers. To understand how the opcodes are constructed refer to the examples below. The API reference manual can be found in the docs folder of the downloadable SDK package provided at mcuxpresso.nxp.com.

MCR and MRC opcodes have several elements that are defined below. The table describes the inputs from the PowerQuad module for each element.

Table 274. MCR and MRC opcodes

Opcode	Parameters
coproc	CP_PQ
opc1	PQ_INV, PQ_LN, PQ_SQRT, PQ_INVSQRT, PQ_ETOX, PQ_ETONX, PQ_DIV, PQ_SIN, PQ_COS, PQ_BIQ0_CALC, PQ_BIQ1_CALC
Rt	(x) – the input value
CRn	PQ_FLOAT32, PQ_FIXEDPT, PQ_COMP0_ONLY, PQ_COMP1_ONLY
CRm	0
opc2	PQ_TRANS, PQ_TRIG, PQ_BIQUAD, PQ_TRANS, PQ_TRIG_FIXED, PQ_BIQUAD_FIXED

Table 275. Parameter descriptions

Parameters	Description
CP_PQ	PowerQuad Module
PQ_INV	Inverse function, 1/x
PQ_LN	Natural logarithm, ln(x)
PQ_SQRT	Square root, sqrt(x)
PQ_INVSQRT	Inverse square root, 1/sqrt(x)
PQ_ETOX	Exponential function, e^(x)
PQ_ETONX	Inverse Exponential function, e^-(x)
PQ_DIV	Division, x1/x2
PQ_SIN	Sine, sin(x)
PQ_COS	Cosine, cos(x)

Table continues on the next page...

Table 275. Parameter descriptions (continued)

Parameters	Description
PQ_BIQ0_CALC	IIR filter for state 0, biquad0(x)
PQ_BIQ1_CALC	IIR filter for state 1, biquad1(x)
PQ_FLOAT32	Floating point data
PQ_FIXEDPT	Fixed-point data
PQ_COMP0_ONLY	Computation Block 0
PQ_COMP1_ONLY	Computation Block 1
PQ_TRANS	Transcendental engine used for floating-point data
PQ_TRIG	Trigonometric engine used for floating-point data
PQ_BIQUAD	Dual Biquad IIR engine used for floating-point data
PQ_TRANS_FIXED	Transcendental engine used for fixed-point data
PQ_TRIG_FIXED	Trigonometric engine used for fixed-point data
PQ_BIQUAD_FIXED	Dual Biquad IIR engine used for fixed-point data

18.6.1.1 Writing to PowerQuad

Coprocessing operations begin when the MCR operation is executed. Since MCR is an Arm operation, Arm assembler instructions are used directly in the application code. Each type of operation contains different elements to execute properly. See below the different operation types and how they are constructed. APIs have already been defined in the SDK for each of the supported mathematical functions. The following is only for reference.

Table 276. Commands

sin(x)
Floating point: __arm_mcr(CP_PQ, PQ_SIN, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRIG)
Fixed point: __arm_mcr(CP_PQ, PQ_SIN, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRIG_FIXED)
cos(x)
Floating point: __arm_mcr(CP_PQ, PQ_COS, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRIG)
Fixed point: __arm_mcr(CP_PQ, PQ_COS, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRIG_FIXED)
ln(x)
Floating point: __arm_mcr(CP_PQ, PQ_LN, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRANS)
Fixed point:

Table continues on the next page...

Table 276. Commands (continued)

__arm_mcr(CP_PQ, PQ_LN, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)
e^(x)
Floating point: __arm_mcr(CP_PQ, PQ_ETOX, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRANS)
Fixed point: __arm_mcr(CP_PQ, PQ_ETOX, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)
e^(-x)
Floating point: __arm_mcr(CP_PQ, PQ_ETONX, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRANS)
Fixed point: __arm_mcr(CP_PQ, PQ_ETONX, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)
1/(x)
Floating point: __arm_mcr(CP_PQ, PQ_INV, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRANS)
Fixed point: __arm_mcr(CP_PQ, PQ_INV, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)
x1/x2
The input for division is calculated with the concatenation of the divider and divisor. x = (x2 | (x1<<32));
Floating point: __arm_mcrr(CP_PQ, PQ_FLOAT32 | PQ_COMP0_ONLY, (uint64_t)(x), PQ_DIV)
1/sqrt(x)
Floating point: __arm_mcr(CP_PQ, PQ_INVSQRT, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRANS)
Fixed point: __arm_mcr(CP_PQ, PQ_INVSQRT, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)
sqrt(x)
Floating point: __arm_mcr(CP_PQ, PQ_SQRT, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRANS)
Fixed point: __arm_mcr(CP_PQ, PQ_SQRT, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)

Table continues on the next page...

Table 276. Commands (continued)

biquad(x)
Floating point: __arm_mcr(CP_PQ, PQ_BIQ0_CALC, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_BIQUAD)
Fixed point: __arm_mcr(CP_PQ, PQ_BIQ0_CALC, (uint32_t)(x), PQ_FIXEDPT | PQ_COMP0_ONLY, 0, PQ_BIQUAD_FIXED)

18.6.1.2 Reading from PowerQuad

To read back the result produced by the above algorithms, use the following MRC opcodes:

Table 277. Read back commands

Floating point:
__arm_mrc(CP_PQ, 0, PQ_FLOAT32 | PQ_COMP0_ONLY, 0, 0) __arm_mrc(CP_PQ, 1, PQ_FLOAT32 | PQ_COMP0_ONLY, 0, 0)
Fixed point:
__arm_mrc(CP_PQ, 0, PQ_FIXEDPT | PQ_COMP0_ONLY, 0, 0) __arm_mrc(CP_PQ, 1, PQ_FIXEDPT | PQ_COMP0_ONLY, 0, 0)

18.6.1.3 Coprocessor API example

18.6.1.3.1 Sinusoidal example

Using a sinusoidal equation as an example, let's build the coprocessing operation to accelerate the time it takes to execute, sin(3.0).

In this first example, the input and output values are floating-point numbers. The input value of 3.0 is in radians.

```
float input = 3.0;
float referenceOutput = 0.141120;
```

As mentioned above, the SDK already provides ready to use APIs for the different mathematical operations. In the case of a sine wave, there are three options that are compatible with CMSIS-DSP operations.

```
arm_sin_f32(float x)
arm_sin_q31(q31_t x)
arm_sin_q15(q15_t x)
```

For floating-point values, the `arm_sin_f32` option is used. This then uses the API from the PowerQuad drivers. The extra variable seen below is used for integer-float conversion during the operation.

```
float32_t arm_sin_f32(float32_t x)
 {
     float tmp;

     PQ_SinF32(&x, &tmp);
     return tmp;
 }
```

Inside the PowerQuad function, we can see that the floating-point variable is converted to an unsigned 32-bit integer, which is what the MCR operation uses to process the input. This is done by using a simple union structure that defines the equivalent

floating-point and integer variable. The blue arrow is the defined function call to the MCR operation constructed in Writing to PowerQuad.

```
static inline void PQ_SinF32(float *pSrc, float *pDst)
 {
      pq_float_t val;

       val.floatX = *pSrc;
      _pq_sin0(val.integerX);
      val.integerX = _pq_readAdd0();
      *pDst       = val.floatX;
 }
```

```
#define _pq_sin0(x) __arm_mcr(CP_PQ, PQ_SIN, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRIG)
```

The blue arrow in the following image is the defined function to the MRC operation constructed in Reading from PowerQuad. This function reads back the result and converts the value back to floating point before storing in the `*pDst` pointer.

```
static inline void PQ_SinF32(float *pSrc, float *pDst)
 {
      pq_float_t val;

       val.floatX = *pSrc;
      _pq_sin0(val.integerX);
      val.integerX = _pq_readAdd0();
      *pDst       = val.floatX;
 }
```

```
#define _pq_readAdd0() __arm_mrc(CP_PQ, 1, PQ_FLOAT32 | PQ_COMP0_ONLY, 0, 0)
```

18.6.1.3.2 Tangent example

Writing to PowerQuad demonstrates that the mapped coprocessing functions for trigonometric applications are sine and cosine. However, using simple math, and other mapped mathematical functions we can build the tangent equation.

Although there are several ways to come to the same tangent solution, note that the chosen equations affect the number of cycles required for the same result. Consider the following three equations:

$$\tan x = \frac{\sin x}{\cos x} \qquad \tan x = \frac{\sin x}{\sqrt{1 - sin^2\ x}} \qquad \tan x = \frac{\cos x}{\sqrt{1 - cos^2\ x}}$$

All three equations provide the same result. However, the two on the right take longer to execute than the one on the left because of the number of cycles required to execute them. Keep in mind that the two on the right are still much faster than doing the same sequence of events without PowerQuad enabled.

For the option on the left, sine and cosine can be done in parallel since these functions use 2 MACs each. Once the values are ready, perform a division. For the two options on the right, square either sine or cosine first, then do the subtraction. Next take the square root, then finally do a division. All of these steps consume many more cycles than the option on the left. Therefore, to realize the full potential of PowerQuad, choose the simplest route when building a new math function based on the available functions.

Since PowerQuad does not natively support the tangent function and we now know which functions to use for efficiency, we can build the code to run on PowerQuad. We want to run two coprocessor functions in parallel, so we must send the MCR opcodes. One uses computation block 0 and the second uses computation block 1. These are already provided in the SDK API, fsl_powerquad.h header file.

```
#define _pq_readAdd0() __arm_mrc(CP_PQ, 1, PQ_FLOAT32 | PQ_COMP0_ONLY, 0, 0)
```

For the purposes of this example, Calculate the tangent of pi/4, which gives us a result of 1, if done correctly. In addition, we use floating-point values as our input and output.

```
float input = (double)(MATH_PI/4.0f);

float finalResult;
```

We use the union structure to convert easily between floating-point and integer values, one for each function. Since the input is the same for the sine and cosine functions used to calculate the tangent, the same value is assigned to both structures.

```
pq_float_t val0, val1, div;

val0.floatX = input;
val1.floatX = input;
```

Then send the commands for each of the opcodes that run on PowerQuad, and the opcodes that read from the resulting registers.

```
_pq_sin0(val0.integerX);
_pq_cos1(val1.integerX);
val0.integerX = _pq_readAdd0();
val1.integerX = _pq_readAdd1();
```

Since the division is done using the intermediate integer values retrieved from both the sine and cosine, they can be used immediately to construct the input of the division and execute the opcode.

```
uint64_t inputDiv = (uint64_t)(val1.integerX) | ((uint64_t)(val0.integerX) << 32U);
_pq_div0(inputDiv);
```

Then read from the resulting register and convert to floating point for the final result.

```
div.integerX = _pq_readMult0();
finalResult = X1.floatX;
```

At this point, the result is 1 for a correctly executed tangent function.

18.6.1.3.3 Biquad cascade example

Having a dual biquad engine allows us to use two different biquad states to execute a chain of second order biquad stages. Where each stage has the same format as described in Dual Biquad IIR. For this example, a 3-stage biquad cascade is shown.

A structure is created to initialize each of the coefficients in each biquad stage. The values vary from application to application, in this case we have the following data. Initialize the COMPREG register with zero, because the register is used for intermediate operations. Importantly, if COMREG has other values, those values can corrupt the result of the operation.

```
pq_biquad_state_t state[3] = {
    {
        .compreg = 0,
        .param =
            {
                .v_n_1 = 0.0f,
                .v_n   = 0.0f,
                .a_1   = 0.2514f,
                .a_2   = 0.5028f,
                .b_0   = 0.2514f,
                .b_1   = -0.1712f,
                .b_2   = 0.1768f,
            },
    },
    {
        .compreg = 0,
        .param =
            {
                .v_n_1 = 0.0f,
                .v_n   = 0.0f,
                .a_1   = 0.2514f,
                .a_2   = 0.5028f,
                .b_0   = 0.2514f,
                .b_1   = -0.1712f,
                .b_2   = 0.1768f,
            },
    },
    {
        .compreg = 0,
        .param =
            {
                .v_n_1 = 0.0f,
                .v_n   = 0.0f,
                .a_1   = 0.2514f,
                .a_2   = 0.5028f,
                .b_0   = 0.2514f,
                .b_1   = -0.1712f,
                .b_2   = 0.1768f,
            },
    },
};
```

The data for the biquad in this example is:

```
float dataForBiquad[16] = {1024.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

There are two possible routes to take when executing a cascaded biquad. The first is to calculate all results at once. The second option is to do incremental calculations.

For the first option, the API provided is:

```
pq_biquad_cascade_df2_instance instance;
PQ_BiquadCascadeDf2Init(&instance, stage, state);
PQ_BiquadCascadeDf2F32(&instance, dataForBiquad, biquadResult, 15);
```

The PQ_BiquadCascadeDf2Init function simply assigns the values for the number of stages in this cascade sequence and the structure of the coefficients for each of those stages. The PQ_BiquadCascadeDf2F32 function takes the input data for the filter and the block size, the result is stored in the output pointer provided.

Internally this function restores and backups the intermediate results produced from each of the states, and performs a biquad operation on each state. This function is done multiple times until all stages have been processed. The Restore and BackUp functions use the GPREG and COMPREG registers for intermediate values.

```
if((stage & 0x01U) == 0x01U)
{
        PQ_BiquadRestoreInternalState(POWERQUAD, 0, states);

        PQ_VectorBiquadDf2F32(pSrc, pDst, (int32_t)blockSize);

        PQ_BiquadBackUpInternalState(POWERQUAD, 0, states);

        states++;
        stage--;
}

while(stage > 0U)
{
        PQ_BiquadRestoreInternalState(POWERQUAD, 1, states);
        states++;
        PQ_BiquadRestoreInternalState(POWERQUAD, 0, states);

        PQ_VectorBiquadCascadeDf2F32(pDst, pDst, (int32_t)blockSize);


        states--;
        PQ_BiquadBackUpInternalState(POWERQUAD, 1, states);
        states++;
        PQ_BiquadBackUpInternalState(POWERQUAD, 0, states);

        states++;
        stage -= 2U;
}
```

When the stage is higher than 0, the operation executed is the PQ_VectorBiquadCascadeDf2F32. This function uses both of the Biquad states. As described in Dual Biquad IIR, a cascade biquad operation uses state 1, then state 0 provides the result. If the stage is 1, then a single biquad operation uses only biquad state 0.

For the incremental option, the function is done with the following APIs:

```
PQ_BiquadCascadeDf2Init(&instance, stage, state);

PQ_BiquadCascadeDf2F32(&instance, dataForBiquad, biquadResult, 8);
PQ_BiquadCascadeDf2F32(&instance, &dataForBiquad[8], &biquadResult[8], 7);
```

The function used for the incremental option is the same as the one used in the full calculation. However, the block size is smaller and partitioned in two segments, with the second function starting where the last one left off. This partitioning is beneficial when you must separate operations and use the PowerQuad for another purpose. You must save the intermediate results to return and run the biquad operation without corrupting the data.

For more details on the internal operations of the biquad filter, review the example code provided in the SDK.

18.6.2 Construct AHB interface operations

Below are some examples demonstrating how registers are configured for some of the supported algorithms in the AHB mechanism. The APIs used in the SDK drivers internally use these configurations, so you do not need to use these configurations directly. To understand how to configure the registers, refer to the examples below. The API reference manual can be found in the docs folder of the downloadable SDK package provided at mcuxpresso.nxp.com.

Understanding the math behind the operation, which needs to be accelerated, facilitates the construction of the AHB interface operation. First, it is necessary to know which algorithm to use, before selecting the correct engine and the proper configurations. In addition to the settings for the algorithm chosen, you must write to the CONTROL register to enable the execution. Writing to the CONTROL register indicates to PowerQuad that it can begin processing the information, therefore write to the CONTROL register last. The CONTROL register is composed of three elements:

INST_BUSY	0: Not Busy 1: Busy

Table continues on the next page...

Table continued from the previous page...

DECODE_MACHINE	1: CP_MTX – Matrix Engine 2: CP_FFT – Transform Engine 3: CP_FIR – Filter Engine 5: CP_CORDIC – Cordic Engine
DECODE_OPCODE	Opcode specific to decode_machine: Matrix: PQ_MTX_SCALE, PQ_MTX_MULT, PQ_MTX_ADD, PQ_MTX_INV, PQ_MTX_PROD, PQ_MTX_SUB, PQ_MTX_DOTP, PQ_MTX_TRANS Transform: PQ_TRANS_CFFT, PQ_TRANS_IFFT, PQ_TRANS_CDCT, PQ_TRANS_IDCT, PQ_TRANS_RFFT, PQ_TRANS_RDCT Filter: PQ_FIR_FIR, PQ_FIR_CONVOLUTION, PQ_FIR_CORRELATION, PQ_FIR_INCREMENTAL Cordic : CORDIC_ARCTAN, CORDIC_ARCTANH, CORDIC_ITER, CORDIC_MIU, CORDIC_T

The following sections describe different AHB interface operations and how they are constructed individually. Each type of operation can use the same registers but in a different manner.

18.6.2.1 Matrix API example

A simple matrix example is the addition of two matrices. Matrix data is stored in memory row-by-row, arranged like standard C arrays.

So, if two 2x2 integer matrices A and B are:

$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

Then the input data is stored in memory arrays as follows:

- int data_A15[4] = {1, 2, 3, 4};
- int data_B15[4] = {5, 6, 7, 8};

In the same manner, we can expect that our resulting matrix has the same format. In matrix addition, the size of the resulting matrix is the same size as the input matrices.

- int Result[4] = {0};

The SDK drivers provide a structure to help organize the matrices. This structure contains key elements of the matrix. It has the number of rows, columns, and the pointer to where the matrix data is stored.

```
typedef struct
{
   uint16_t numRows;    /**< number of rows of the matrix.     */
   uint16_t numCols;    /**< number of columns of the matrix.  */
   q15_t  *pData;       /**< points to the data of the matrix. */
   arm_matrix_instance_q15;
}
```

Using this structure the three matrices that are needed can be initialized. Since the rows and columns are the same for all of them, we can reuse the variable. The initialization shown below simply assigns the value of the columns and rows to their respective variable in the structure, and points the matrix data to the pointer of the structure.

```
srcRows    = MATRIX_ROWS;
srcColumns = MATRIX_COL;
arm_mat_init_q15(&addMatrixA15, srcRows, srcColumns, A15);
arm_mat_init_q15(&addMatrixB15, srcRows, srcColumns, B15);
arm_mat_init_q15(&addMatrixR, srcRows, srcColumns, Result);
```

For compatibility, we can use the CMSIS-DSP function call directly, which calls the PowerQuad APIs when enabled. It uses the three matrices that are needed for the addition operation.

```
arm_mat_add_q15(&addMatrixA15, &addMatrixB15, &addMatrixR);
```

This example uses fixed-point 16-bit inputs and outputs. For this we must configure the OUTFORMAT, INAFORMAT, INBFORMAT, and TMPFORMAT registers. The drivers facilitate this configuration by providing a defined framework.

```
#define PQ_SET_FIX16_CONFIG                                                                                \
    POWERQUAD->OUTFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_16Bit << 4U) | (uint32_t)kPQ_Float; \
    POWERQUAD->INAFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_16Bit << 4U) | (uint32_t)kPQ_Float; \
    POWERQUAD->INBFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_16Bit << 4U) | (uint32_t)kPQ_Float; \
    POWERQUAD->TMPFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_Float << 4U) | (uint32_t)kPQ_Float; \
    POWERQUAD->TMPBASE   = 0xE0000000U
```

All of the FORMAT registers are configured in the same manner. No scaler is necessary. The output format is fixed 16-bit data and the internal engine format is handled using floating point. Also, the TMPBASE is defined as 0xE000_0000, which is the reference for private RAM.

Next, it is necessary to define the value of LENGTH. As described in Requirements, LENGTH is remapped depending on the operation. In this case, we need the number of columns for matrix B15 and both the columns and rows for matrix A15.

- LENGTH[31:0] = Cols_B[20:16] | Cols_A[12:8] | Rows_A[4:0]

So, LENGTH = 0x40404;

As described above, the CMSIS-DSP function, internally calls the PowerQuad function.

```
arm_status arm_mat_add_q15(const arm_matrix_instance_q15 *pSrcA,
                           const arm_matrix_instance_q15 *pSrcB,
                           arm_matrix_instance_q15 *pDst)

{
    assert(NULL != pSrcA);
    assert(NULL != pSrcB);
    assert(NULL != pDst);

    arm_status status;
    uint32_t length;

    PQ_SET_FIX16_CONFIG;

    length = POWERQUAD_MAKE_MATRIX_LEN(pSrcA->numRows, pSrcA->numCols, pSrcB->numCols);

    PQ_MatrixAddition(POWERQUAD, length, pSrcA->pData, pSrcB->pData, pDst->pData);

    /* Wait for the completion */
    PQ_WaitDone(POWERQUAD);

    status = ARM_MATH_SUCESS;

    /* Return to application */
    return status;
}
```

The API that configures the registers for matrix addition is as follows:

- PQ_MatrixAddition(POWERQUAD, length, data_A, data_B, result);

Table 278. PQ_MatrixAddition parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
length	Composed of the length of the M2 column, M1 row and column. When M1 and M2 are matrix A and B respectively. Length = (length[23:16] = M2 columns | length [15:8] = M1 columns | length [7:0] = M1 rows);
data_A	Base address where the first matrix is located
data_B	Base address where the second matrix is located
result	Base address where the result matrix is expected to be located

The register configurations are set as shown below:

- PowerQuad[OUTBASE] = address of Result;
- PowerQuad[INABASE] = address of A15;
- PowerQuad[INBBASE] = address of B15;
- PowerQuad[LENGTH] = length;
- PowerQuad[CONTROL] = (CP_MTX << 4U) | PQ_MTX_ADD;

Lastly, it is important that the EVENTEN register is enabled to trigger when execution is completed. Events and Interrupts are generally set up during initialization of PowerQuad.

```
arm_status arm_mat_add_q15(const arm_matrix_instance_q15 *pSrcA,
                const arm_matrix_instance_q15 *pSrcB,
                arm_matrix_instance_q15 *pDst)

{
        assert(NULL != pSrcA);
        assert(NULL != pSrcB);
        assert(NULL != pDst);

        arm_status status;
        uint32_t length;

        PQ_SET_FIX16_CONFIG;

        length = POWERQUAD_MAKE_MATRIX_LEN(pSrcA->numRows, pSrcA->numCols, pSrcB->numCols);

        PQ_MatrixAddition(POWERQUAD, length, pSrcA->pData, pSrcB->pData, pDst->pData);

        /* Wait for the completion */
        PQ_WaitDone(POWERQUAD);

        status = ARM_MATH_SUCESS;

        /* Return to application */
        return status;
}
```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```
/* wait for the completion */
while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
        _WFE();
}
```

At this point, the result can be read from the Result matrix that we initialized at the beginning of the example.

18.6.2.2 FFT API example

The Complex Fast Fourier Transform(CFFT) is a different AHB interface example that uses the transform engine. There are certain requirements and details that must be considered when using the FFT engine. See Requirements for more details.

This example uses fixed 32-bit format data. The input data is taken by using PowerQuad functions and using sine and cosine waves. The input data can be generated from many different sources. For ease of use, the example uses the PowerQuad module.

```
/* Two full period sin wave. */
for (uint32_t i = 0; i < CFFT_INPUT_LEN / 2;  i++)
{
        input[i * 2]     = (arm_sin_q31(i * (0x80000000 / (CFFT_INPUT_LEN / 2))) / 2) >> 5;
        input[i * 2 + 1] = (arm_cos_q31(i * (0x80000000 / (CFFT_INPUT_LEN / 2))) / 2) >> 5;
}
```

Once the input data is ready, PowerQuad must be configured accordingly. For compatibility we can use the CMSIS-DSP function call directly, which calls the PowerQuad APIs when enabled.

```
arm_cfft_q31(&instance, input, 0, 1);
```

This example uses fixed-point 32-bit inputs and outputs. For this we must configure the OUTFORMAT, INAFORMAT, INBFORMAT, and TMPFORMAT registers. The drivers facilitate this configuration by providing a defined framework.

```
#define PQ_SET_FFT_Q31_CONFIG                                                   \
    POWERQUAD->OUTFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
    POWERQUAD->INAFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
    POWERQUAD->INBFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
    POWERQUAD->TMPFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
    POWERQUAD->TMPBASE   = 0xE0000000U
```

All of the FORMAT registers are configured in the same manner. There is no scaler necessary. The output format is fixed 32-bit data and the internal engine format is handled using fixed point as well. Also, the TMPBASE is defined as 0xE000_0000, which is the reference for private RAM.

As described above, the CMSIS-DSP function, internally calls the PowerQuad function.

```
void arm_cfft_q31(const arm_cfft_instance_q31 *S, q31_t *p1, uint8_t ifftFlag, uint8_t bitReverseFlag)
{
    assert(bitReverseFlag == 1U);

    q31_t *pIn     = p1;
    q31_t *pOut    = p1;
    uint32_t length = S->fftLen;

    PQ_SET_FFT_Q31_CONFIG;

    if (ifftFlag == 1U)
    {
        PQ_TransformIFFT(POWERQUAD, length, pIn, pOut);
    }
    else
    {
        PQ_TransformCFFT(POWERQUAD, length, pIn, pOut);
    }

    PQ_WaitDone(POWERQUAD);
}
```

The API that configures the registers for CFFT is as follows:

- PQ_TransformCFFT(POWERQUAD, N, inputData, cfftResult);

Table 279. PQ_TransformCFFT parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
N	Length for FFT and DCT, the supported lengths are 16, 32, 64, 128, 256, and 512
inputData	Base address where the input array is located
cfftResult	Base address where the result matrix is expected to be located

The register configurations are set as shown below:

- PowerQuad[OUTBASE] = address of Result;
- PowerQuad[INABASE] = address of input;
- PowerQuad[LENGTH] = N;
- PowerQuad[CONTROL] = (CP_FFT << 4U) | PQ_TRANS_CFFT;

Lastly, it is important that the EVENTEN register is enabled to trigger when execution is completed. Events and Interrupts are generally set up in the initialization of the PowerQuad.

```
void arm_cfft_q31(const arm_cfft_instance_q31 *S, q31_t *p1, uint8_t ifftFlag, uint8_t bitReverseFlag)
{
        assert(bitReverseFlag == 1U);

        q31_t *pIn   = p1;
        q31_t *pOut   = p1;
        uint32_t length = S->fftLen;

        PQ_SET_FFT_Q31_CONFIG;

        if (ifftFlag == 1U)
        {
                PQ_TransformIFFT(POWERQUAD, length, pIn, pOut);
        }
        else
        {
                PQ_TransformCFFT(POWERQUAD, length, pIn, pOut);
        }

        PQ_WaitDone(POWERQUAD);
}
```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```
/* wait for the completion */
 while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
        _WFE();
}
```

At this point, the result can be read from the Result matrix that we initialized at the beginning of the example.

18.6.2.3 FIR API example

The FIR engine can compute different processes. One of these signal processing mechanisms is convolution. Convolution is one of the most important and fundamental concepts in signal processing and analysis.

Shown here is the mathematical definition of convolution in a discrete time domain.

$$pDst[n] = pSrcA[n] * pSrcB[n] = \sum_{k=-\infty}^{\infty} pSrcA[k] \bullet pSrcB[n-k]$$

Where `pSrcA [n]` is the input signal, `pSrcB[n]` is the impulse response, and `y[n]` is the output.

For this example, the data used is defined as floating-point values, however this example uses 32-bit fixed-point data.

```
static const float pSrcA[] = {0.01f, 0.02f, 0.03f, 0.04f, 0.05f};
static const float pSrcB[] = {0.02f, 0.04f, 0.06f, 0.08f, 0.10f};
```

The drivers in the SDK provide an easy API to convert from floating-point to fixed-point 32-bit data.

```
inputA[i] = FLOAT_2_Q31(pSrcA[i]);
inputB[i] = FLOAT_2_Q31(pSrcB[i]);
```

Once the input data is defined, the PowerQuad must be configured accordingly. For compatibility, we can use the CMSIS-DSP function call directly, which calls the PowerQuad APIs when enabled.

```
arm_conv_q31(inputA, CONV_A_LEN, inputB, CONV_B_LEN, result);
```

The data uses fixed-point 32-bit inputs and outputs. For this, we must configure the OUTFORMAT, INAFORMAT, INBFORMAT, and TMPFORMAT registers. The drivers facilitate this configuration by providing a defined framework.

```
#define PQ_SET_Q31_CONFIG                                                   \
        POWERQUAD->OUTFORMAT = ((uint32_t)(-31) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_Float; \
        POWERQUAD->INAFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_Float;   \
        POWERQUAD->INBFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_Float;   \
        POWERQUAD->TMPFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_Float << 4U) | (uint32_t)kPQ_Float;   \
        POWERQUAD->TMPBASE   = 0xE0000000U
```

All of the FORMAT registers are configured in the same manner, except OUTFORMAT and TMPFORMAT. The output format is scaled for 32 bits. In addition, the output and the input formats are fixed 32-bit data. The internal engine format is handled using floating point. The temporary format remains floating point throughout its use. Also, the TMPBASE is defined as 0xE000_0000, which is the reference for private RAM.

As described above, the CMSIS-DSP function, internally calls the PowerQuad function.

```
void arm_conv_q31(const q31_t *pSrcA, uint32_t srcALen, const q31_t *pSrcB, uint32_t srcBLen, q31_t *pDst)
{
    assert(NULL != pSrcA);
    assert(NULL != pSrcB);
    assert(NULL != pDst);

    PQ_SET_Q31_CONFIG;

    PQ_FIR(POWERQUAD, pSrcA, (int32_t)srcALen,  pSrcB, (int32_t)srcBLen, pDst,  PQ_FIR_CONVOLUTION);
    PQ_WaitDone(POWERQUAD);
}
```

The API that configures the registers for FIR is as follows:

- PQ_FIR(POWERQUAD, pSrcA, srcALen, pSrcB, srcBLen, pDst, op_type);

Table 280. PQ_FIR parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
pSrcA	Base address where the input signal is located
srcALen	Length of the input signal
pSrcB	Base address where the impulse response signal is located
srcBLen	Length of the impulse response
pDst	Base address where the result is expected to be located
op_type	Filter operation that is the computer

The register configurations are set as shown below:

- PowerQuad[OUTBASE] = address of pDst;

- PowerQuad[INABASE] = address of pSrcA;
- PowerQuad[INBBASE] = address of pSrcB;
- PowerQuad[LENGTH] = (srcBLen << 16) | srcALen;
- PowerQuad[CONTROL] = (CP_FIR << 4U) | PQ_FIR_CONVOLUTION;

Enable the EVENTEN register to trigger when execution is completed. Set up Events and Interrupts during PowerQuad initialization.

```
void arm_conv_q31(const q31_t *pSrcA, uint32_t srcALen, const q31_t *pSrcB, uint32_t srcBLen, q31_t *pDst)
{
        assert(NULL != pSrcA);
        assert(NULL != pSrcB);
        assert(NULL != pDst);

        PQ_SET_Q31_CONFIG;

        PQ_FIR(POWERQUAD, pSrcA, (int32_t)srcALen,  pSrcB, (int32_t)srcBLen, pDst,  PQ_FIR_CONVOLUTION);
        PQ_WaitDone(POWERQUAD);
}
```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```
/* wait for the completion */
 while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
        _WFE();
}
```

The results are read from the result pointer, initialized in the configuration registers.

18.6.3 CORDIC API examples

The inverse trigonometric functions are implemented using a CORDIC. PowerQuad implements two-iteration per cycle CORDIC. The return value is in the range of -2^27 to 2^27, which represents -pi to pi. The sum of data_A and data_B should not exceed the range of INT32.

This example demonstrates the arctangent of 0.5. Larger input numbers gets higher output accuracy, the result of PQ_ArctanFixed(POWERQUAD, 200000, 100000, kPQ_Iteration_24) is more accurate than PQ_ArctanFixed(POWERQUAD, 2, 1, kPQ_Iteration_24).

The PowerQuad functions can be used directly as there is not support for the arctangent function in the CMSIS-DSP library. The input and output data are handled as 32-bit fixed-point data.

```
PQ_ArctanFixed(DEMO_POWERQUAD, 20000000, 10000000, kPQ_Iteration_24);
```

In this example, the FORMAT registers are not used. Instead we directly provide the inputs to the CORDIC registers of the Powerquad. CORDIC_X contains the dividend and CORDIC_Y contains the divisor. CORDIC_Z contains the result once the execution has finished. To initialize this register write 0.

```
int32_t pq_arctan(int32_t x int32_t y)
{
int32_t result = PQ_ArctanFixed(DEMO_POWERQUAD, x, y, kPQ_Iteration_24);
        result += (y<0 && x<0 )? 0xf8000000;
                  (y>=0 && x<0 )? 0x08000000;

return (result);
}
```

The API that configures the registers for arctan using CORDIC is as follows:

- PQ_ArctanFixed(POWERQUAD, data_A, data_B, iteration);

Table 281. PQ_ArctanFixed parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
data_A	Base address where the value of the opposite side
data_B	Base address where the value of the adjacent side
iteration	Value of iterations for rotations in the calculation can be 8, 16, or 24

The register configurations are set as shown below:

- PowerQuad[CORDIC_X] = data_A;
- PowerQuad[CORDIC_Y] = data_B;
- PowerQuad[CORDIC_Z] = 0;
- PowerQuad[CONTROL] = (CP_CORDIC << 4U) | CORDIC_ARCTAN | CORDIC_ITER(iteration);

Enable the EVENTEN register to trigger when execution is completed. Set up Events and Interrupts during PowerQuad initialization.

```
int32_t PQ_ArctanFixed(POWERQUAD_Type *base, int32_t x, int32_t y, pq_cordic_iter_t iteration)
{
    base->CORDIC_X = (uint32_t)x;
    base->CORDIC_Y = (uint32_t)y;
    base->CORDIC_Z = 0U;
    base->CONTROL  = (CP_CORDIC << 4U) | CORDIC_ARCTAN | CORDIC_ITER(iteration);

    PQ_WaitDone(base);
    return (int32_t)base->CORDIC_Z;
}
```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```
/* wait for the completion */
while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
        _WFE();
}
```

The result can be read from the CORDIC Input Z (CORDIC_Z) register.

18.7 Memory map and register definition

This section includes the PowerQuad module memory map and detailed descriptions of all registers.

NOTE

This module does not generate transfer errors when accessing unimplemented registers in its register space.

18.7.1 PowerQuad register descriptions

18.7.1.1 PowerQuad memory map

POWERQUAD0 base address: 400B_F000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Output Base (OUTBASE)	32	RW	0000_0000h
4h	Output Format (OUTFORMAT)	32	RW	0000_0000h
8h	Temporary Base (TMPBASE)	32	RW	0000_0000h
Ch	Temporary Format (TMPFORMAT)	32	RW	0000_0000h
10h	Input A Base (INABASE)	32	RW	0000_0000h
14h	Input A Format (INAFORMAT)	32	RW	0000_0000h
18h	Input B Base (INBBASE)	32	RW	0000_0000h
1Ch	Input B Format (INBFORMAT)	32	RW	0000_0000h
100h	Control (CONTROL)	32	RW	0000_0000h
104h	Length (LENGTH)	32	RW	0000_0000h
108h	Coprocessor Prescale (CPPRE)	32	RW	0000_0000h
10Ch	Miscellaneous (MISC)	32	RW	0000_0000h
110h	Cursory (CURSORY)	32	RW	0000_0000h
180h	CORDIC Input X (CORDIC_X)	32	RW	0000_0000h
184h	CORDIC Input Y (CORDIC_Y)	32	RW	0000_0000h
188h	CORDIC Input Z (CORDIC_Z)	32	RW	0000_0000h
18Ch	Error Status (ERRSTAT)	32	RW	0000_0000h
190h	Interrupt Enable (INTREN)	32	RW	0000_0000h
194h	Event Enable (EVENTEN)	32	RW	0000_0000h
198h	Interrupt Status (INTRSTAT)	32	RW	0000_0000h
200h - 23Ch	General Purpose Register Bank n (GPREG0 - GPREG15)	32	RW	0000_0000h
240h - 25Ch	Compute Register Bank n (COMPREG0 - COMPREG7)	32	RW	0000_0000h

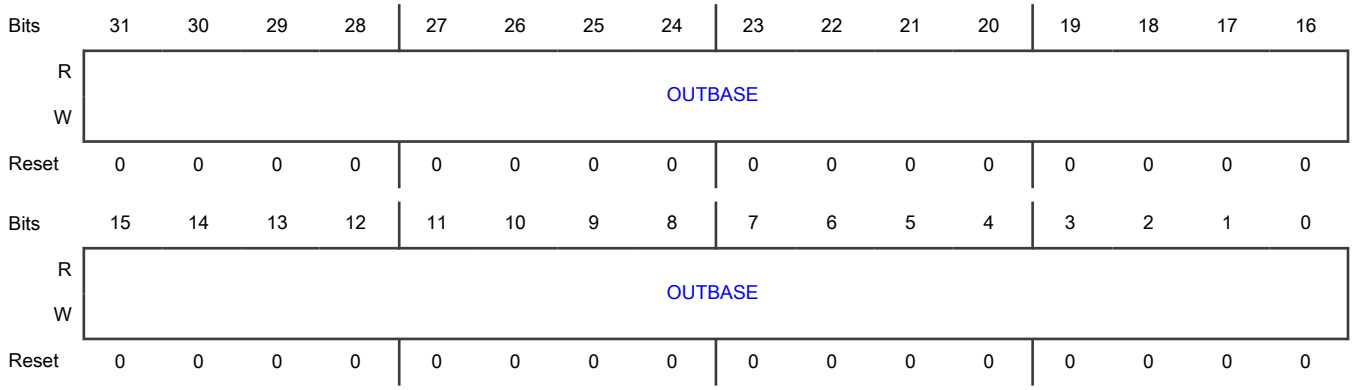
18.7.1.2 Output Base (OUTBASE)

Offset

Register	Offset
OUTBASE	0h

Function
Contains the base address, which points to a location in memory where you want to store the result of the operation.

Diagram



Fields

Field	Function
31-0 OUTBASE	Output Region Base Address 32-bit register that contains the base address where the result is stored upon completion of the AHB operation.

18.7.1.3 Output Format (OUTFORMAT)

Offset

Register	Offset
OUTFORMAT	4h

Function

Used to define the format of the data inside the engine and outside the engine in the OUTBASE location. The scaling factor can also be chosen for fixed-point output.

NOTE

The format definitions can only be used when data regions are in AHB space. Data written to or read from private RAM space does not undergo conversion.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OUT_SCALER								Reserved		OUT_FORMAT EXT		Reserved		OUT_FORMATI NT	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-8 OUT_SCALER	8-bit Scaling Value for Result Data Used when converting from floating=point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 OUT_FORMAT EXT	Output External Format Format of result data, converted when exiting the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0 OUT_FORMATI NT	Output Internal Format Format of result data during active operation within the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved

18.7.1.4 Temporary Base (TMPBASE)

Offset

Register	Offset
TMPBASE	8h

Function

Used to define the location in which temporary data can be stored while operations are being executed. PowerQuad uses its own private RAM for this purpose. The base address for this area is 0xE000_0000.

NOTE

Inverse matrix operations do not use the private RAM area. The TMPBASE address must point to a scratch area in AHB space.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TMPBASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TMPBASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 TMPBASE	Base Address for the Temporary Region Contains the base address for the temporary data stored during some operations.

18.7.1.5 Temporary Format (TMPFORMAT)

Offset

Register	Offset
TMPFORMAT	Ch

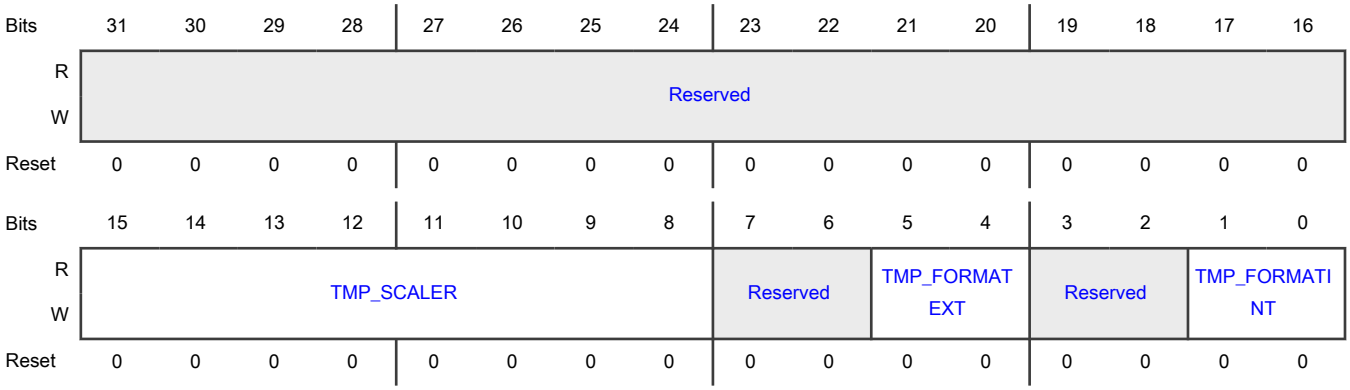
Function

Used to define the format of the data inside the engine and outside the engine in the TMPBASE location. The scaling factor can also be chosen for fixed-point output.

NOTE

The format definitions can only be used when data regions are in AHB space. Data written to or read from private RAM space does not undergo conversion.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 TMP_SCALER	Scaling Value for Temporary Data. 8-bit scaling value used when converting from floating-point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 TMP_FORMAT EXT	Temporary External Format Format of temporary data, converted when exiting the engine. **NOTE** This conversion can only be used for temporary data in AHB space, it cannot be used for temporary data stored in private RAM. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0	Temporary Internal Format Format of temporary data during active operation within the engine.

Table continues on the next page...

Table continued from the previous page...

Field	Function
TMP_FORMATINT	00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved

18.7.1.6 Input A Base (INABASE)

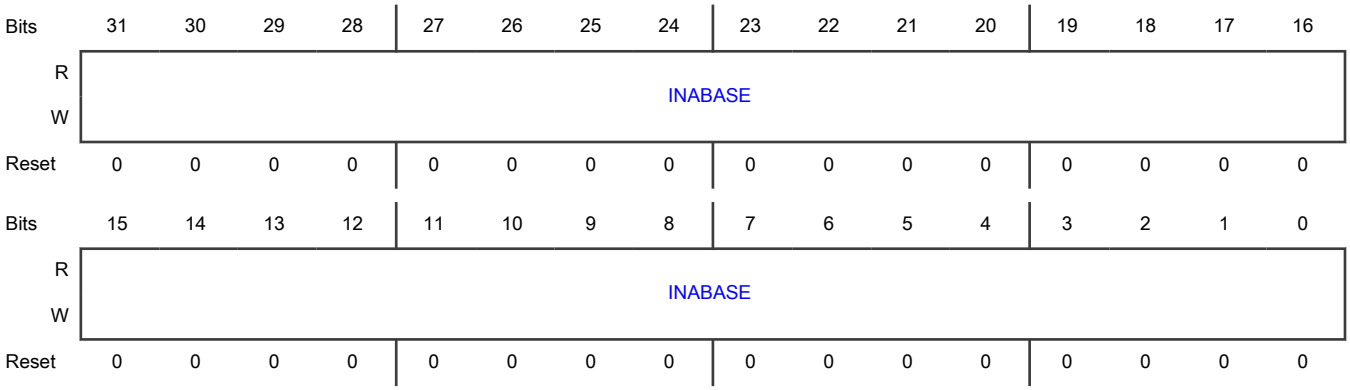
Offset

Register	Offset
INABASE	10h

Function

Contains the base address that points to a location in memory where you store operand A.

Diagram



Fields

Field	Function
31-0 INABASE	Input A Base Contains the base address where operand A is stored prior to executing the AHB operation.

18.7.1.7 Input A Format (INAFORMAT)

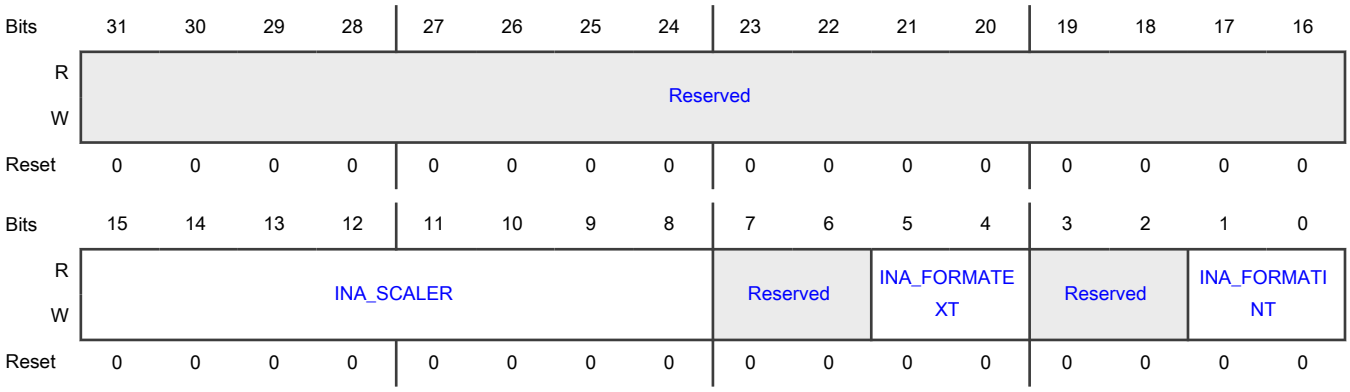
Offset

Register	Offset
INAFORMAT	14h

Function

Used to define the format of the data inside the engine and outside the engine in the INABASE location. The scaling factor can also be chosen for fixed-point output.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 INA_SCALER	Input A Scaler Value 8-bit scaling value for operand A data. Used when converting from floating-point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 INA_FORMATEXT	Input A External Format Original format of operand A data prior to entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0 INA_FORMATINT	Input A Internal Format Format of operand A data, converted when entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - F32 32-bit floating-point format 11b - Reserved

18.7.1.8 Input B Base (INBBASE)

Offset

Register	Offset
INBBASE	18h

Function

Contains the base address that points to a location in memory where you store operand B.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INBBASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INBBASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 INBBASE	Input B Base Contains the base address for where operand B is stored prior to executing the AHB operation.

18.7.1.9 Input B Format (INBFORMAT)

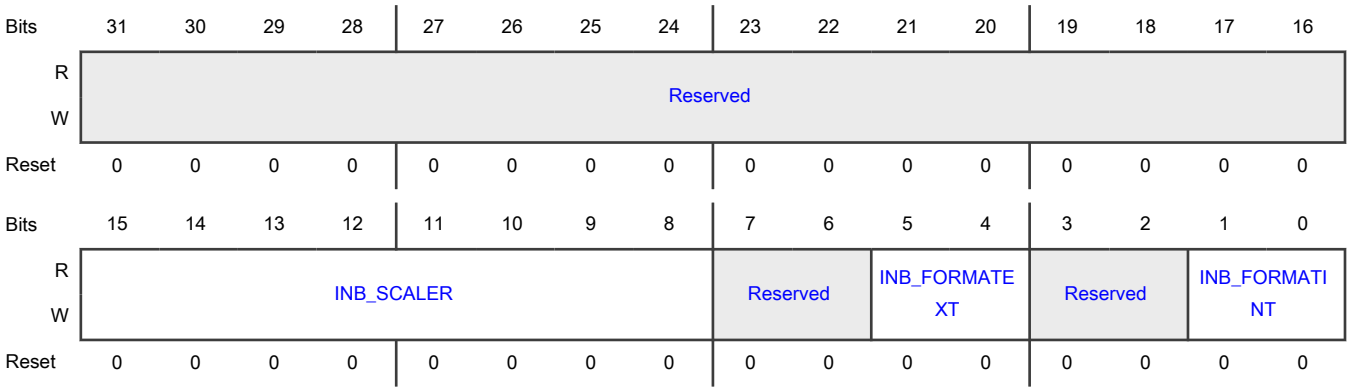
Offset

Register	Offset
INBFORMAT	1Ch

Function

Used to define the format of the data inside the engine and outside the engine in the INBBASE location. The scaling factor can also be chosen for fixed-point output.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 INB_SCALER	Input B Scaler 8-bit scaling value for operand B data. Used when converting from floating-point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 INB_FORMATEXT	Input B External Format Original format of operand B data prior to entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0 INB_FORMATINT	Input B Internal Format Format of operand B data conversion when entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - F32 32-bit floating-point format 11b - Reserved

18.7.1.10 Control (CONTROL)

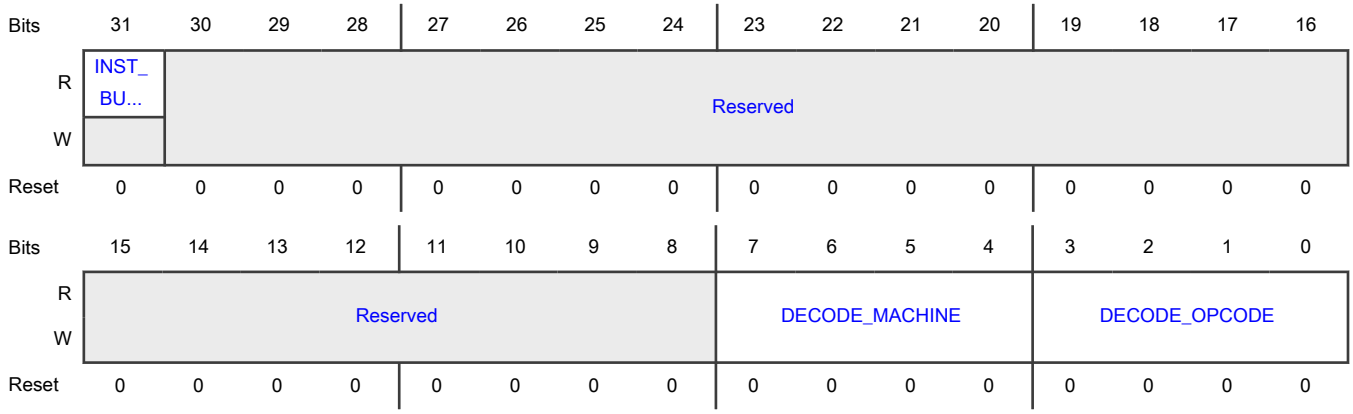
Offset

Register	Offset
CONTROL	100h

Function

Used for AHB operations. You can define the executable operations. When writing to the CONTROL register, the operation begins. A status bit is available for polling to determine if PowerQuad is busy.

Diagram



Fields

Field	Function
31 INST_BUSY	Instruction Busy PowerQuad is busy performing requested operations. 0b - Not busy 1b - Busy
30-8 —	Reserved
7-4	Decode Machine Computation engine to be used for operation.

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>DECODE_MACHINE</td><td>0000b - Coprocessor
0001b - Matrix engine
0010b - Transform engine
0011b - Filter engine
0100b,0110b-1111b - Reserved
0101b - CORDIC engine</td></tr>
<tr><td>3-0
DECODE_OPCODE</td><td>Decode Opcode
Defines what operation that the engine executes. Each engine can of different operations. The table below describes possible operations for each engine:
Table 282. Decode Opcodes
<table>
<tr><th></th><th>CP_MTX</th><th>CP_FFT</th><th>CP_FIR</th><th>CP_CORDIC</th></tr>
<tr><td>0000b</td><td>-</td><td>PQ_TRANS_CFFT</td><td>PQ_FIR_FIR</td><td>-</td></tr>
<tr><td>0001b</td><td>PQ_MTX_SCALE</td><td>PQ_TRANS_IFFT</td><td>PQ_FIR_CONVOLUTION</td><td>CORDIC_ARCTAN</td></tr>
<tr><td>0010b</td><td>PQ_MTX_MULT</td><td>PQ_TRANS_CDCT</td><td>PQ_FIR_CORRELATION</td><td>-</td></tr>
<tr><td>0011b</td><td>PQ_MTX_ADD</td><td>PQ_TRANS_IDCT</td><td>-</td><td>CORDIC_ARCTANH</td></tr>
<tr><td>0100b</td><td>PQ_MTX_INV</td><td>PQ_TRANS_RFFT</td><td>PQ_FIR_INCREMENTAL</td><td>-</td></tr>
<tr><td>0101b</td><td>PQ_MTX_PROD</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>0110b</td><td>-</td><td>PQ_TRANS_RDCT</td><td>-</td><td>-</td></tr>
<tr><td>0111b</td><td>PQ_MTX_SUB</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>1000b</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>1001b</td><td>PQ_VEC_DOTP</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>1010b</td><td>PQ_MTX_TRAN</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>1011b - 1111b</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
</table></td></tr>
</table>

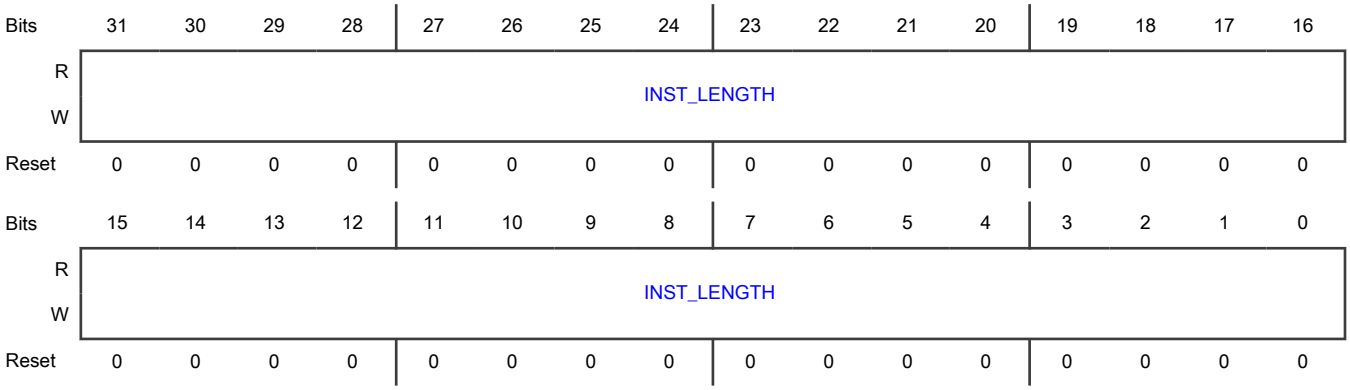
18.7.1.11 Length (LENGTH)

Offset

Register	Offset
LENGTH	104h

Function
Length register

Diagram



Fields

Field	Function

31-0
INST_LENGTH
	Instruction length
	The INST_LENGTH field is remapped according to the operating engine in use. The value of this field provides the dimensions of the operands.
	For Matrix Operations:
	`LENGTH[31:0] = Cols_B[20:16] | Cols_A[12:8] | Rows_A[4:0]`
	For FIR Operations:
	`LENGTH[31:0] = numTaps[31:16] | numSamples[15:0]`
	For other operations:
	`LENGTH[31:0] is the value of size of the input operand.`
	NOTE
	Maximum row/column is 16 by 16, regardless of 5-bit allowance.
	For vector operations, the maximum size is 256 (vector operations have only one dimension for length).

18.7.1.12 Coprocessor Prescale (CPPRE)

Offset

Register	Offset
CPPRE	108h

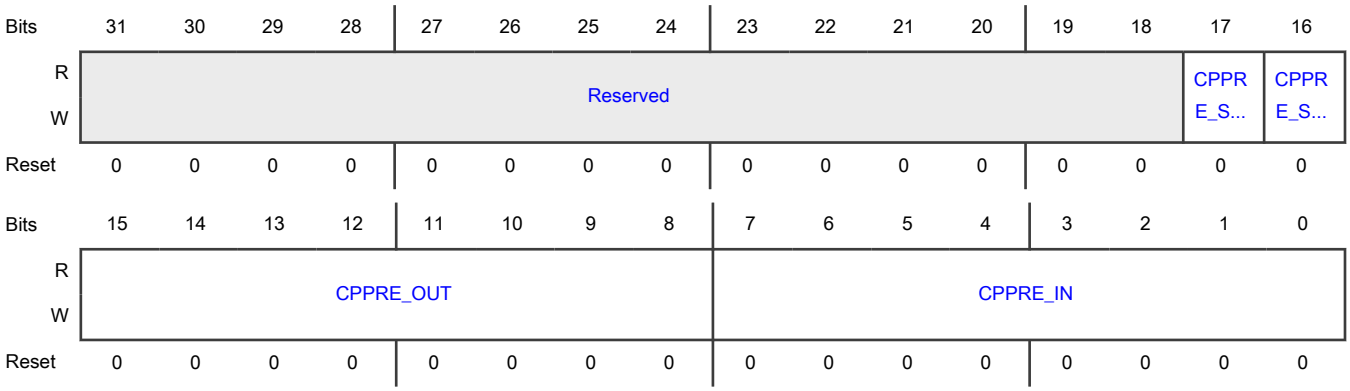
Function

Used to define the format of the input and output data used via the coprocessor bus interface. In addition, you can set the saturation for the output.

NOTE

CPPRE does not affect data used via the AHB bus interface.

Diagram



Fields

Field	Function
31-18 —	Reserved
17 CPPRE_SAT8	Saturation 8 Output value saturation is applied at 'n' bits, when 'n' is either 8 bits or 16 bits. 0b - 8 bits 1b - 16 bits
16 CPPRE_SAT	Saturation Sub-32 bit saturation. Enforces saturation for the output data. If enabled, CPPRE_SAT8 defines saturation. 0b - No saturation 1b - Forces sub-32 bit saturation
15-8 CPPRE_OUT	Postscaling Output Contains the coprocessor postscaling value for fixed-point output data with fractional precision.
7-0 CPPRE_IN	Prescaling Input Contains the coprocessor scaling value for fixed-point input data with fractional precision.

18.7.1.13 Miscellaneous (MISC)

Offset

Register	Offset
MISC	10Ch

Function

Used to store the scaling value used for matrix scale operations. Each element of the matrix scales by the value of this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INST_MISC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INST_MISC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 INST_MISC	Scaling Factor 32-bit value that scales the elements of a matrix.

18.7.1.14 Cursory (CURSORY)

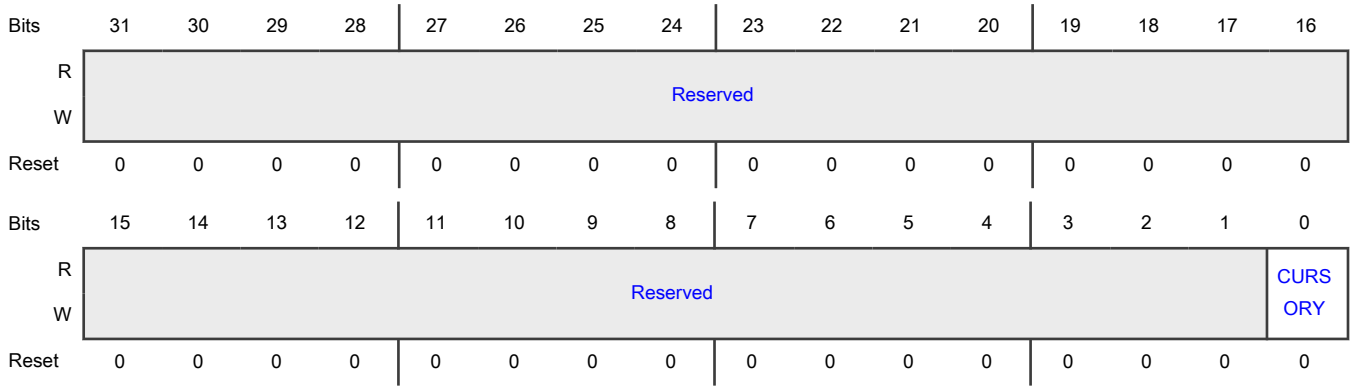
Offset

Register	Offset
CURSORY	110h

Function

Enables PowerQuad to reduce the resolution of results by zeroing out the LSB bits of the input and output of the floating-point MACs. Reducing the resolution reduces power consumption. Use when PowerQuad does not require greater accuracy.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 CURSORY	Cursory Mode Cursory mode allows control of the input and output resolution to the MACs used by PowerQuad engines. 0b - Disable cursory mode, full floating-point accuracy (24-bit mantissa + 2 bits before rounding). 1b - Enable cursory Mode, 16-bit mantissa (bottom bits are zeroed for inputs and outputs of MACs).

18.7.1.15 CORDIC Input X (CORDIC_X)

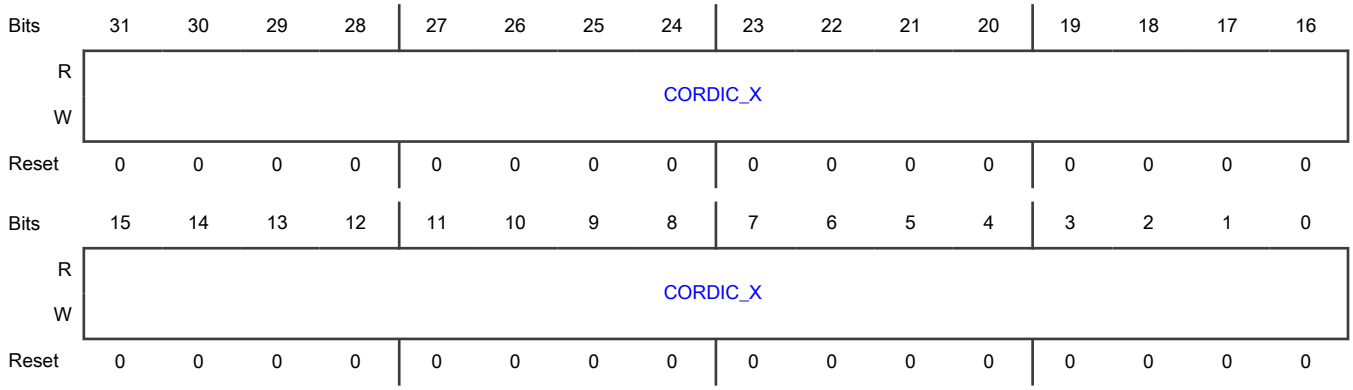
Offset

Register	Offset
CORDIC_X	180h

Function

Used for CORDIC-based algorithms. Contains the value of the first operand in the operation.

Diagram



Fields

Field	Function
31-0 CORDIC_X	CORDIC Input X Fixed-point operand, X, for CORDIC-based operations.

18.7.1.16 CORDIC Input Y (CORDIC_Y)

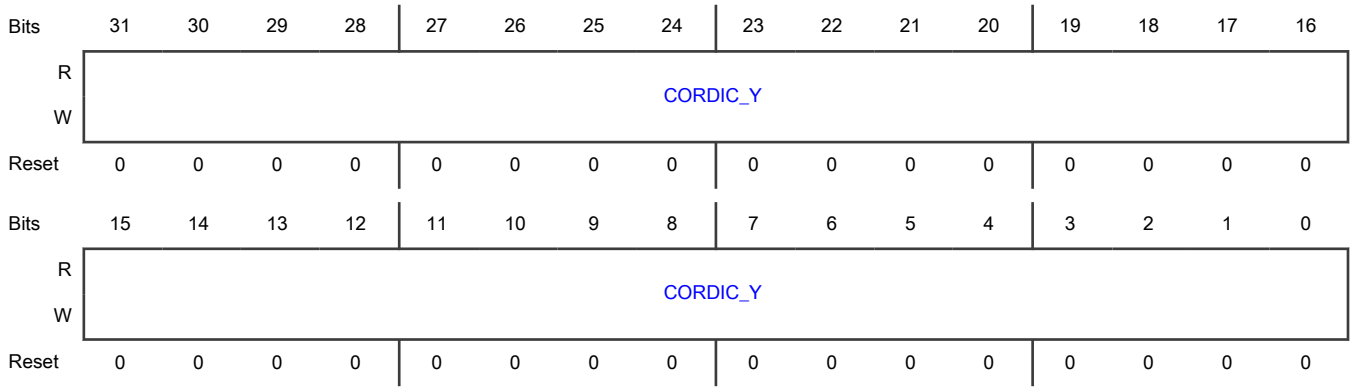
Offset

Register	Offset
CORDIC_Y	184h

Function

Used for CORDIC-based algorithms. Contains the value of the second operand in the operation.

Diagram



Fields

Field	Function
31-0 CORDIC_Y	CORDIC Input Y Fixed-point operand, Y, for CORDIC-based operations.

18.7.1.17 CORDIC Input Z (CORDIC_Z)

Offset

Register	Offset
CORDIC_Z	188h

Function

Used for CORDIC-based algorithms. Contains the value of the result in the operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CORDIC_Z															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CORDIC_Z															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CORDIC_Z	CORDIC Input Z Fixed-point result data, Z, for CORDIC-based operations.

18.7.1.18 Error Status (ERRSTAT)

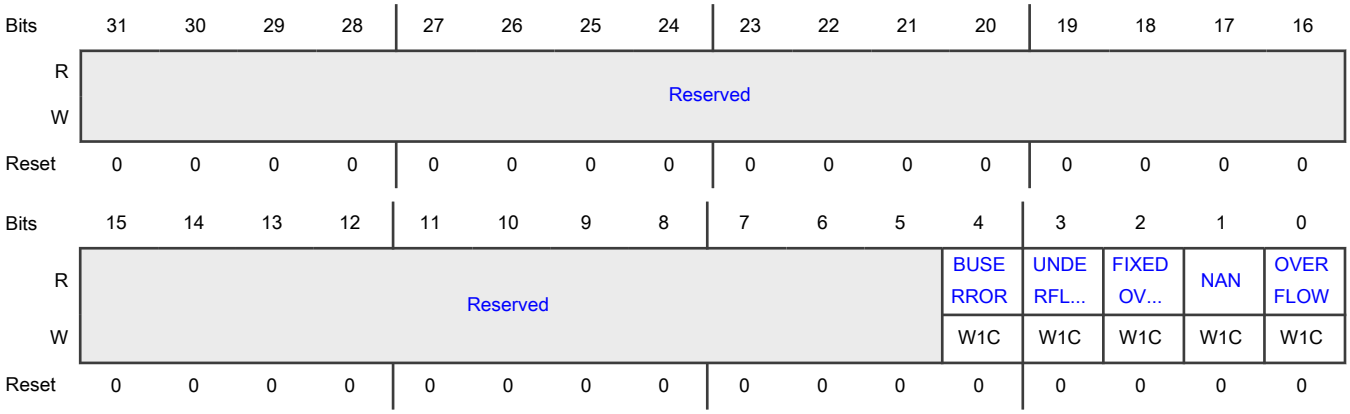
Offset

Register	Offset
ERRSTAT	18Ch

Function

Provides 5 sticky status bits that capture an error during a PowerQuad operation. All bits are 'Write 1 to Clear'(W1C).

Diagram



Fields

Field	Function
31-5 —	Reserved
4 BUSERROR	Bus Error An AHB master interface bus error has occurred. 0b - No error 1b - Error on bus
3 UNDERFLOW	Underflow Error occurs when the input is so small that it has been truncated to 0. 0b - No error 1b - Error on underflow
2 FIXEDOVERFLOW	Fixed-point Overflow Error occurs when there is more data than can be processed. This bit refers to fixed-point values. 0b - No error 1b - Error on fixed-point overflow
1 NAN	Floating-Point Not-a-Number (NaN) Error occurs when the data has been corrupted or is illegal. 0b - No error 1b - Error on floating-point NaN
0	Floating-point Overflow

Table continues on the next page...

Table continued from the previous page...

Field	Function
OVERFLOW	Error occurs when there is more data than can be processed. This bit refers to floating-point values. 0b - No error 1b - Error on floating-point overflow

18.7.1.19 Interrupt Enable (INTREN)

Offset

Register	Offset
INTREN	190h

Function

Determines which conditions trigger an interrupt, based on the set status of the ERRSTAT register and on successful completion of operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								INTR_CO...	Reserved		INTR_BE...	INTR_UF...	INTR_FI...	INTR_NAN	INTR_OF...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7 INTR_COMP	Interrupt on Instruction Completion 0b - Disable interrupt 1b - Enable interrupt
6-5 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 INTR_BERR	Interrupt on AHBM Bus Error 0b - Disable interrupt 1b - Enable interrupt
3 INTR_UFLOW	Interrupt on Underflow 0b - Disable interrupt 1b - Enable interrupt
2 INTR_FIXED	Interrupt on Fixed-point Overflow 0b - Disable interrupt 1b - Enable interrupt
1 INTR_NAN	Interrupt Floating-point NaN 0b - Disable interrupt 1b - Enable interrupt
0 INTR_OFLOW	Interrupt Floating-point Overflow 0b - Disable interrupt 1b - Enable interrupt

18.7.1.20 Event Enable (EVENTEN)

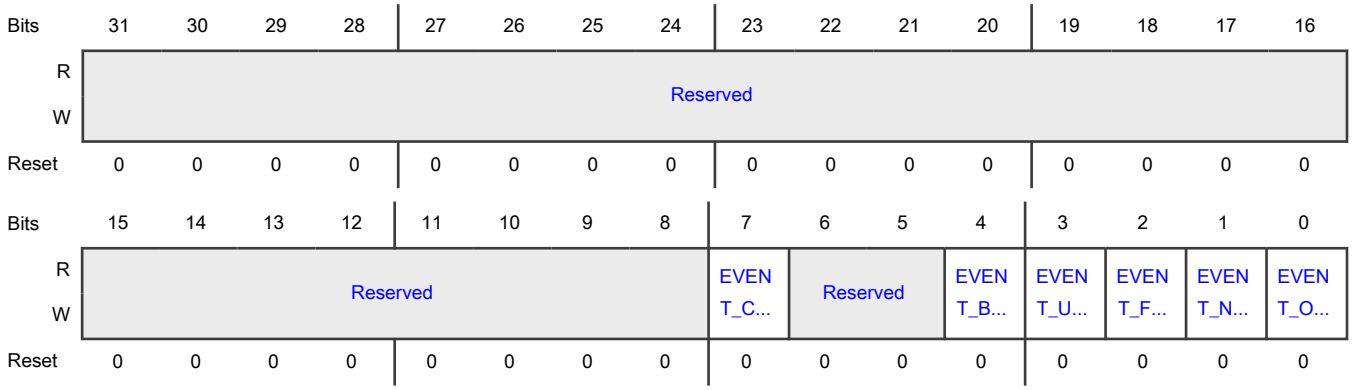
Offset

Register	Offset
EVENTEN	194h

Function

Determines which conditions trigger an event based on the set status of the ERRSTAT register and on successful completion of operation.

Diagram



Fields

Field	Function
31-8 —	Reserved
7 EVENT_COMP	Event Trigger on Instruction Completion 0b - Disable event trigger 1b - Enable event trigger
6-5 —	Reserved
4 EVENT_BERR	Event Trigger on AHBM Bus Error 0b - Disable event trigger 1b - Enable event trigger
3 EVENT_UFLOW	Event Trigger on Underflow 0b - Disable event trigger 1b - Enable event trigger
2 EVENT_FIXED	Event Trigger on Fixed-point Overflow 0b - Disable event trigger 1b - Enable event trigger
1 EVENT_NAN	Event Trigger on Floating-Point NaN 0b - Disable event trigger 1b - Enable event trigger
0 EVENT_OFLOW	Event Trigger on Floating-point Overflow 0b - Disable event trigger 1b - Enable event trigger

18.7.1.21 Interrupt Status (INTRSTAT)

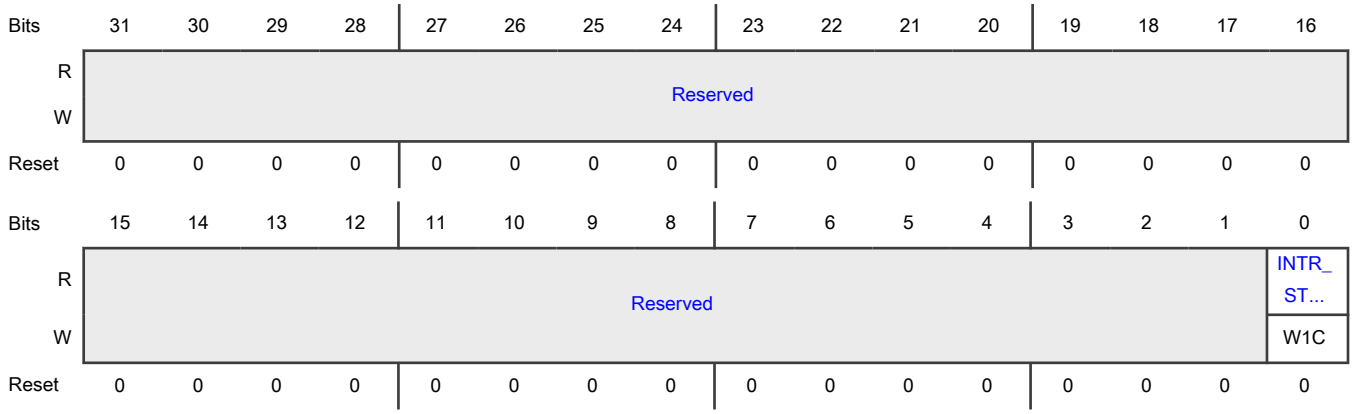
Offset

Register	Offset
INTRSTAT	198h

Function

Notifies on asserted trigger condition. A write of any value clears the Interrupt Status bit.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 INTR_STAT	Interrupt Status 0b - No new interrupt 1b - Interrupt captured

18.7.1.22 General Purpose Register Bank n (GPREG0 - GPREG15)

Offset

For n = 0 to 15:

Register	Offset
GPREGn	200h + (n × 4h)

Function

The General Purpose Register Bank is n x 32 bits. Used within the PowerQuad MAC engines. The values seen here are the staging and result registers of the adders and multipliers. These banks are made accessible so that you can back up the registers, when needed, to use PowerQuad for a different thread, then restore these saved values.

These registers are used by the Dual Biquad IIR engine and the Transform engine for intermediate data.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPREG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPREG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 GPREG	General Purpose Bank Intermediate staging and result data can be pulled and restored as needed. However, it is important to consider when data is being manipulated, because it can corrupt the results of an operation.

18.7.1.23 Compute Register Bank n (COMPREG0 - COMPREG7)

Offset

For n = 0 to 7:

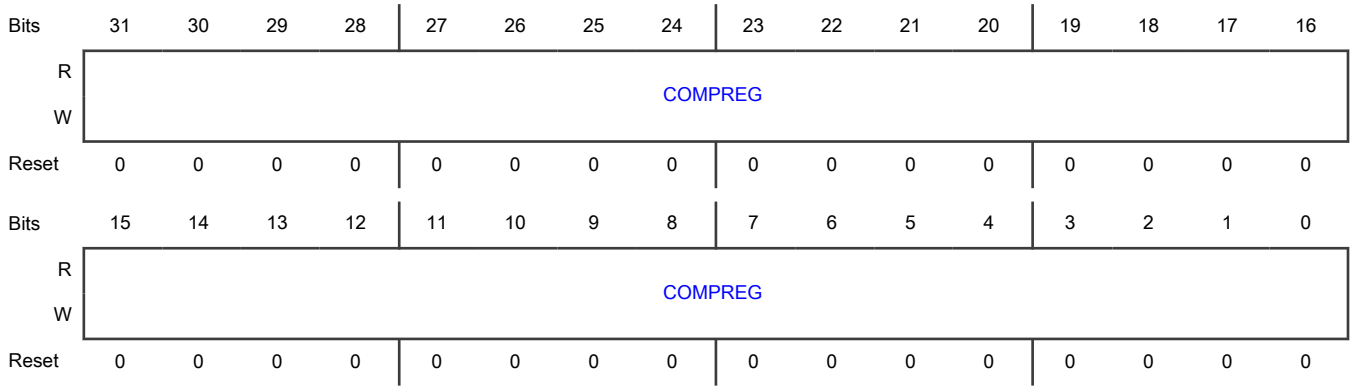
Register	Offset
COMPREGn	240h + (n × 4h)

Function

The Compute Register Bank is n x 32 bits. Used within the PowerQuad MAC engines. The values seen here are the staging or intermediate data used in FFT or Biquad operations. These banks are made accessible so that you can back up the registers, when needed, to use PowerQuad for a different thread. Then restore these saved values.

These registers are only used by the Dual Biquad IIR engine.

Diagram



Fields

Field	Function
31-0 COMPREG	Compute bank Intermediate staging and result data can be pulled and restored as needed. However it is important to consider when data is being manipulated, because it can corrupt the results of an operation.

Chapter 19
SmartDMA Controller

19.1 Chip-specific SmartDMA Controller information

Table 283. Reference links to related information

Topic	Related module	Reference
Full description	SmartDMA Controller	SmartDMA Controller
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See SMARTDMAARCHB_INMUX0 - SMARTDMAARCHB_INMUX7 registers in INPUTMUX

19.1.1 Clock

The SmartDMA clock can be enabled or disabled by setting the SmartDMA field in the AHBCLKCTRL1 register.

19.1.2 Reset

The SmartDMA module can be reset by setting the corresponding field in the PRESETCTRL1 register. Clear the SmartDMA peripheral reset using the PRESETCTRLCLR1 register before using SmartDMA.

19.1.3 Interrupts

SmartDMA interrupt number is 53 (SmartDMA_IRQ).

19.1.4 Triggers

SmartDMA has 71 input sources in total in this chip. The following table lists the trigger sources of SmartDMA:

Table 284. SmartDMA trigger sources

Input	Function
0	GPIO P0_0
1	GPIO P0_1
2	GPIO P0_2
3	GPIO P0_3
4	GPIO P0_4
5	GPIO P0_5
6	GPIO P0_6
7	GPIO P0_7
8	GPIO P0_8
9	GPIO P0_9

Table continues on the next page...

Table 284. SmartDMA trigger sources (continued)

Input	Function
10	GPIO P0_10
11	GPIO P0_11
12	GPIO P0_12
13	GPIO P0_13
14	GPIO P0_14
15	GPIO P0_15
16	SCT_OUT8
17	SCT_OUT9
18	Reserved
19	Reserved
20	MRT_CH0_IRQ
21	MRT_CH1_IRQ
22	CTIMER4_MAT3
23	CTIMER4_MAT2
24	CTIMER3_MAT3
25	CTIMER3_MAT2
26	CTIMER1_MAT3
27	CTIMER1_MAT2
28	UTICK_IRQ
29	WDT0_IRQ
30	ADC0_IRQ
31	CMP0_IRQ
32	Reserved
33	LP_FLEXCOMM7_IRQ
34	LP_FLEXCOMM6_IRQ
35	LP_FLEXCOMM5_IRQ
36	LP_FLEXCOMM4_IRQ
37	LP_FLEXCOMM3_IRQ
38	LP_FLEXCOMM2_IRQ
39	LP_FLEXCOMM1_IRQ
40	LP_FLEXCOMM0_IRQ
41	DMA0_IRQ

Table continues on the next page...

Table 284. SmartDMA trigger sources (continued)

Input	Function
42	DMA1_IRQ
43	SYS_IRQ[1]
44	RTC_COMBO_IRQ
45	ARM_TXEV
46	PINT GPIO_INT_BMAT
47	Reserved
48	Reserved
49	CMP0_OUT
50	USB0 start of frame
51	USB1 start of frame
52	OS_EVENT_TIMER_IRQ
53	ADC1_IRQ
54	CMP0_IRQ/CMP1_IRQ/CMP2_IRQ
55	DAC0_IRQ
56	DAC1_IRQ/DAC2_IRQ
57	PWM0_IRQ
58	PWM1_IRQ
59	QDC0_IRQ
60	QDC1_IRQ
61	EVTG_OUT0A
62	EVTG_OUT1A
63	Reserved
64	Reserved
65	GPIO1 Pin Event Trig 0
66	GPIO1 Pin Event Trig 1
67	GPIO2 Pin Event Trig 0
68	GPIO2 Pin Event Trig 1
69	GPIO3 Pin Event Trig 0
70	GPIO3 Pin Event Trig 1

1. SYS_IRQ combines the CDOG IRQ, WWDT IRQ, MBC secure violation IRQ, Secure AHB Matrix secure violation IRQ, GDET IRQ, ELS S50 error IRQ, PKC error IRQ, and VBAT IRQ using the logical OR operation.

19.1.5 Memory map

The priority of AHB bus masters can be adjusted by using the SYSCON[AHBMATPRIO] register.

19.2 Overview

SmartDMA is a core that supports unique, reduced instruction sets. It works in a similar way to the Arm core. Being the controller of AHB matrix, SmartDMA can access:

- All its targets.
- The GPIO peripheral control and data registers.

The purpose of SmartDMA is to perform event- and I/O-driven handling to offload the Arm processor in the system.

19.2.1 Block diagram

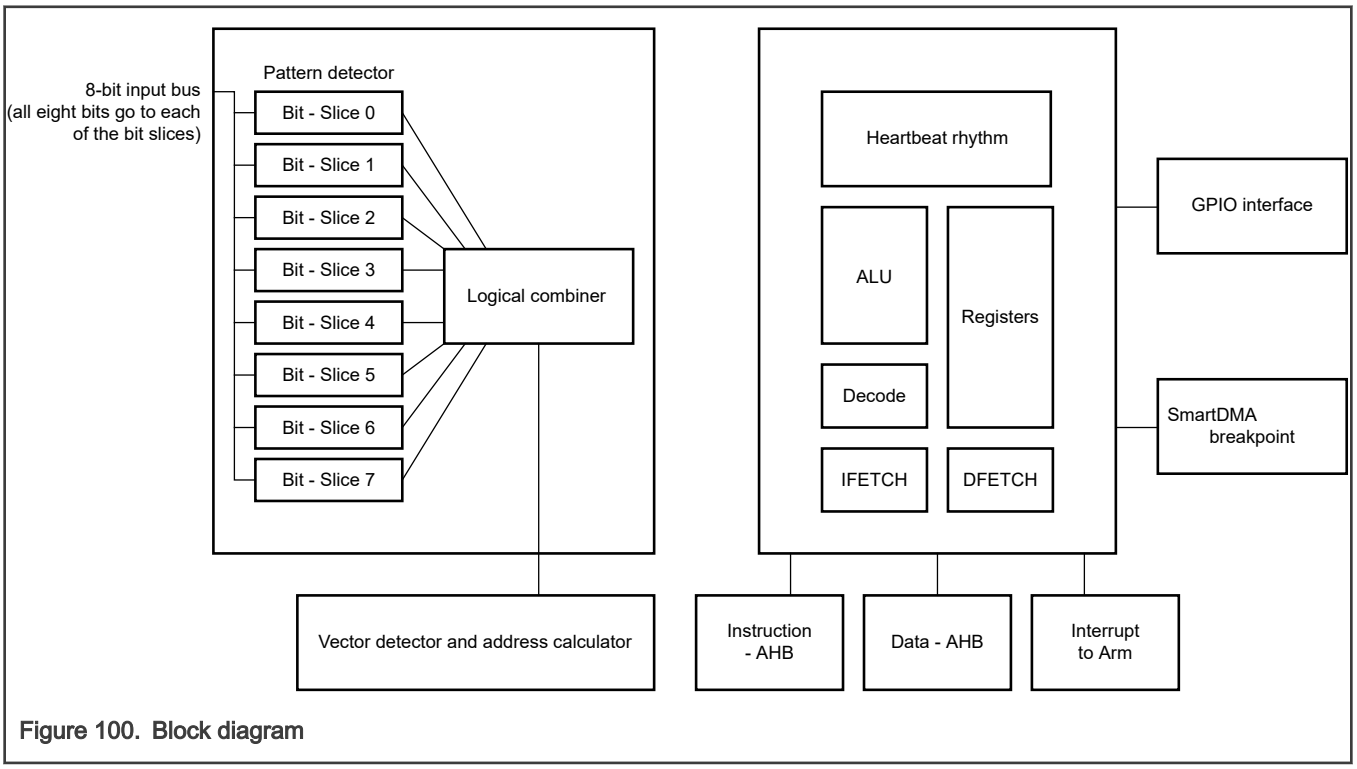


Figure 100. Block diagram

19.2.2 Features

SmartDMA is designed to deliver an optimal performance for the following purposes:

- State machine control
- Boolean detection
- Tedious or repetitive simple tasks
- DMA function
- Interrupt load reduction on Arm
- Streaming GPIO
- General I/O handling
- Protocol emulation over I/O, such as parallel camera and LCD interfaces
- Shift-based algorithm calculation
- Large-scale data manipulation

19.3 Functional description

19.3.1 Clocking

SmartDMA uses the same system clock as the Arm core does. You can enable or disable the SmartDMA clock by configuring the appropriate register field (see the chip-specific section for details). However, you must enable the clock for SmartDMA before setting the SmartDMA controller and SmartDMA program execution.

19.3.2 Reset

You can reset the SmartDMA module by configuring the corresponding register field. See the chip-specific section for details.

19.3.3 Interrupts

You can enable the Arm interrupt for SmartDMA via nested vectored interrupt controller (NVIC). See the chip-specific section for details.

19.3.4 Pin description

SmartDMA can access and manipulate a maximum of 32 I/O pins that are selected as “SmartDMA function”. You can configure “SmartDMA function” by using signal multiplexing.

SmartDMA includes dedicated GPIO control registers through which it can quickly access I/O ports. See the chip-specific section for details.

19.3.5 Requests and triggers

In general, SmartDMA requests set the pace of execution to match what the peripheral (including its FIFO if it has one) could do. For example,

- USART issues a transmit SmartDMA request when its transmit FIFO is not full, and a receive SmartDMA request when its receive FIFO is not empty.
- The GPIO pin issues a data storage SmartDMA request when its rising edge is detected.
- The timer event issues a PWM output SmartDMA request when the counter reaches zero.
- ADC issues a result reading SmartDMA request after the conversion is complete.

SmartDMA includes a number of channels connected to external input sources. See the chip-specific section for details.

19.3.6 Setting and power consumption in different modes

In Sleep mode, SmartDMA:

- Is active.
- Can operate and access all enabled peripherals and memories.
- Can wake up the Arm core from Sleep mode.

In Power-Down and Deep Power-Down modes, SmartDMA:

- Is inactive.
- Is configurable for low-power consumption by:
  - Disabling the SmartDMA clock when not in use.
  - Placing SmartDMA into the reset state.
  - Holding the SmartDMA core by using some opcodes.
  - Slowing down the frequency of SmartDMA execution.

19.4 Memory map

SmartDMA includes two AHB bus master interfaces. You can adjust the priority of AHB bus masters to achieve the desired system performance. See the chip-specific section for details.

Chapter 20
Neural Processing Unit (NPU)

20.1 Chip-specific NPU information

Table 285. Reference links to related information

Topic	Related module	Reference
Full description	NPU	NPU
System memory map		Memory map
Power management		Power management

NOTE

Clock speed of the NPU is derived directly from the system_clk.

20.2 Overview

This chapter provides information on the features supported by the eIQ® Neutron N1-16 NPU. For enablement details and constraints, see the documentation that is part of eIQ Toolkit at http://nxp.com/eiq.

The NPU is an on-chip hardware accelerator that provides efficient processing of machine-learning workloads. It works in conjunction with the chip's CPU to deliver higher performance and lower power consumption for neural network (NN) inference.

In addition to managing Multiply-Accumulate (MAC) compute blocks throughput, NPU also:

- Optimizes arrangement of weights
- Manages memory accesses
- Provides rescaling, activation, and pooling functions

Enablement support for the eIQ Neutron N1-16 NPU is provided with the eIQ Toolkit and eIQ Machine Learning (ML) Software Development Environment. Some of the features and operators listed in this chapter may not be enabled in eIQ software. See References for links to eIQ documentation that lists the features and operators enabled in the latest eIQ software.

You can import your trained and quantized NN into eIQ Toolkit. The eIQ tooling generates an NPU-optimized model that can be integrated into an ML application running on the target device.

20.2.1 Block diagram

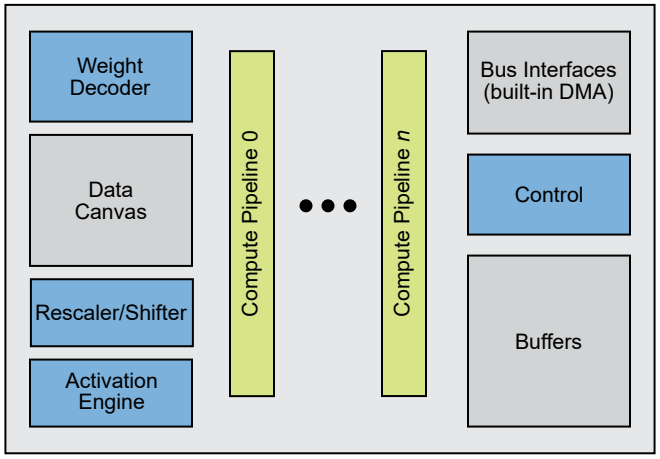


Figure 101. eIQ Neutron NPU architecture

Table 286.

Component	Description
Activation Engine	Provides activation functions. See the list of supported activation functions.
Buffers	Data buffers for pipeline input data and MAC results.
Bus Interfaces	Provide Interconnect fabric designed for efficient data transfer between memories and compute blocks.
Control	Provides control and status functions.
Data Canvas	Provides 2D circular data buffer that enables data reuse among compute pipelines.
Rescaler/Shifter	Provides post-scaling of MAC compute block results. See Rescale support.
Weight Decoder	Manages decompression, if necessary, for weight fetches. See Weight decode support.

20.2.2 Features

- Performance:
  - Up to 16 MACs/cycle for 8-bit × 8-bit operations
- Supported data types:
  - Input data:
    - 8-bit unsigned asymmetric
    - 8-bit signed asymmetric
    - 8-bit signed symmetric
  - Weights:
    - 8-bit signed symmetric
    - 8-bit unsigned symmetric
  - Arithmetic operations:
    - 8-bit data × 8-bit weight

  - Activation × Activation (transformers, for example):
    - 8-bit symmetric × 8-bit symmetric (signed or unsigned)
  - Bias:
    - Custom FP16 format (1-bit sign, 5-bit up-shift, 10 bits significant value)
  - Results:
    - 32-bit symmetric signed
    - 16-bit symmetric signed/unsigned
    - 8-bit symmetric/asymmetric signed/unsigned
- Supported operators: see Operator summary.
- Constraints for Spatial Compute (Convolution, for example):
  - Padding. Individually selectable for Top/Bottom/Left/Right; maximum padding is 4 pixels.
  - Stride. 1 or 2 pixels (horizontal and vertical).
  - Dilation. 1 or 2 pixels (horizontal and vertical).
  - Kernel size. Maximum weight matrix row length = 64 k.
  - Kernel height. Maximum = 8 pixels (Includes pixels skipped due to dilation).
  - Kernel width. Maximum = 64 pixels (Includes pixels skipped due to dilation).

NOTE

Larger kernel-shapes, stripes, and so on, can be achieved through cascading multiple layers.

- Accumulation support:

  eIQ Neutron N1-16 NPU uses a 32-bit accumulation scheme, which prevents saturation—even in large neural networks—for 8-bit × 8-bit computations.
- Weight decode support:
  - 8-bit weights direct (no decode)
  - Mini-weights (<8 bits) through sign extension
  - Mini-weights (≤6 bits). Weights are translated using a codebook. You can use weight clustering to have fewer used 8-bit values and then encode the chosen values by using an index table. NPU uses the weights as indices to select appropriate 8-bit values as the weights are loaded into compute.
- Rescale support:

  Rescales 32-bit accumulator results.
  - In a typical case in which the quantizer can rely on the integer precision dynamic range to perform the ReLU clipping and max value clipping, this function scales values to the desired 8-bit or 16-bit output directly.
  - Rescaling is also used to scale the accumulator result so it can be presented as 16-bit data to NPU's Activation Engine.

  The general form of the rescaling stage is:

```
Output = Round((accumulator × scaleFactor)>>downshiftFactor)
```

  eIQ Neutron N1-16 NPU supports the following rescaling scheme:
  - 16-bit rescale factor: 10-bit scaler, 6-bit downshift factor
- Activation Engine:

eIQ Neutron N1-16 NPU provides built-in activation functions and a hardware engine to perform activation. The Activation Engine works on 16-bit signed data produced by rescaling.

The activation functions are executed “in-pipeline” together with the computation preceding it. No extra cycles are consumed and there is no need for spilling higher precision data into memory than the activated data size.

Activation engine accuracy is 8-bit precision, except for ReLu, which essentially has no modification to values, and exp(x), which is output as 16-bit data.

The supported built-in activation functions are:

- 8-bit:
  - ReLU
  - ReLUx
  - Sigmoid
  - Tanh
  - LeakyReLU (limited to Alpha = 0.25, 0.125, and 0.0625)

20.3 Functional description

20.3.1 Operator summary

Table 287. Operator summary

Operator[1]	TFLite operator	Data size (bits)		Notes
		Input[2]	Output[3]	
Absolute	ABS	8	8	
Add Tensor-Tensor	ADD	8	8	Includes scaling to match quantization scales; both inputs must be of equal bit width
Add Tensor + Constant	ADD	8	8	
Average Pooling	AVERAGE_POOL_2D	8	8	
Concatenate Tensors	CONCATENATE	8	8	
Convolution 2D/1D	CONV_2D	8	8	
Depthwise Convolution 2D/1D	DEPTHWISE_CONV_2D	8	8	
Expand Dimensions	EXPAND_DIMS			Supported by Activate Function Interpolator
Fully Connected / Dense / Linear	FULLY_CONNECTED	8	8	Support for 8- or 16-bit weights and data of 8- or 16-bit data
Leaky-ReLU	LEAKY_RELU	8	8	• As fused activation function or TFLite operator LEAKY_RELU • 8-bit precision for scaling LeakyReLU data
Logistic / Sigmoid	LOGISTIC	8	8	As fused activation or TFLite operator LOGISTIC

Table continues on the next page...

Table 287. Operator summary (continued)

Operator[1]	TFLite operator	Data size (bits)		Notes
		Input[2]	Output[3]	
Maximum	MAXIMUM	8	8	
Max-Pooling	MAX_POOL_2D	8	8	
Mean / Average	MEAN	8	8	
Minimum	MINIMUM	8	8	
Multiplication	MUL, BatchMatMul	8	8	• Support for all types of multiplications—batched, non-batched, dot-product matrix-matrix, matrix-scalar, transpose matrix-matrix, Hadamard • 8-bit data × 8-bit weight
Packing	PACK	8	8	
Padding	PAD	8	8	Implicit with compute
Re-Quantize	QUANTIZE	8	8	• To change INT quantization scale and zero point • Integer input, integer output
RELU	RELU	8	8	As fused activation or TFLite operator RELU
RELU6	RELU6	8	8	• As fused activation or TFLite operator RELU6 • Supports min-max clipping factors other than (0, 6)
RELU	RELU_N1_TO_1	8	8	As fused activation or TFLite operator RELU_N1_TO_1
Reshape Tensor	RESHAPE	8	8	
Resize Bilinear	RESIZE_BILINEAR	8	8	Constraint: Only 2×2. Larger resize factors can be achieved by cascading multiple 2×2 resize layers.
Resize Nearest Neighbor	RESIZE_NEAREST_NEIGHBOR	8	8	Constraint: Only 2×2. Larger resize factors can be achieved by cascading multiple 2×2 resize layers.
Shape	SHAPE	N/A		No operation is performed by NPU hardware
Slice Tensor	SLICE, STRIDED_SLICE	8	8	

Table continues on the next page...

Table 287. Operator summary (continued)

Operator[1]	TFLite operator	Data size (bits)		Notes
		Input[2]	Output[3]	
Split Tensor	SPLIT, SPLIT_V	8	8	
Squeeze Tensor	SQUEEZE			Operation is not performed by NPU hardware
Strided Slice	STRIDED_SLICE	8	8	
Subtract Tensor-Tensor	SUB	8	8	• Includes scaling to match quantization scales • 8-bit data × 8-bit weight
Subtract Tensor-Constant	SUB	8	8	• Includes scaling to match quantization scales • 8-bit data × 8-bit weight
Tanh / hyperbolic tangent function	TANH	8	8	As fused activation or TFLite operator TANH
Transpose Convolution 2D	TRANSPOSE_CONV	8	8	Includes possibility of upsampling input (zero-filled) before operation
Unidirectional LSTM	UNIDIRECTIONAL_SEQUENCE_LSTM	8	8	
Unpack Tensor	UNPACK			No operation is performed by NPU hardware

1. Some of the operators in listed in this table may not be enabled in eIQ software. See References for links to eIQ documentation that lists the operators enabled in the latest eIQ software.
2. Inputs are always 8-bit.
3. Output can be 8-,16-, or 32-bit. Note that this is a format—it does not imply 32-bit precision, but output can be generated in int32 format with whatever data is generated. This can be useful if you want to run, for example, a convolution with the intent of running activation outside of Neutron. It is also sometimes useful to export results as 16-bit data to have symmetric data instead of 8-bit asymmetric data.

20.3.2 References

Topic	Link
eIQ ML Software Development Environment	http://www.nxp.com/eiq
eIQ Toolkit	https://www.nxp.com/design/software/eiq-ml-development-environment/eiq-toolkit-for-end-to-end-model-development-and-deployment:EIQ-TOOLKIT

20.4 Memory map and register definitions

eIQ Neutron NPU does not have any user-configurable registers. All NPU configuration and control functions must be performed using NXP's eIQ Toolkit software. See References for links.

Chapter 21
CoolFlux BSP32 Subsystem

21.1 Chip-specific CoolFlux BSP32 information

Table 288. Reference links to related information

Topic	Related module	Reference
Full description	CoolFlux BSP32	CoolFlux BSP32
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

21.1.1 Module instances

This device has one instance of the CoolFlux BSP32 module, BSP32_0.

21.1.2 RAM

CoolFlux program memory interface can access flash and FlexSPI also along with RAMX, RAMA ~ RAMH. CoolFlux X data memory interface can also access mailbox and Powerquad RAM.

21.1.3 Compiler options

Due to long memory paths from data memory output directly to data memory address, the maximum frequency that can be obtained is limited. The compiler should disable the sequence that consists of a data memory load to a pointer register, followed in the next cycle by a memory access using that same pointer.

21.2 Overview

The CoolFlux BSP32 is a programmable digital signal processor, where BSP stands for Baseband Signal Processor and the 32 indicates that is has a 32-bit data path and 32-bit data memories. It is dual Harvard, dual multiplier processor based on the ultra-low power CoolFlux architecture. It supports 16 and 32-bit scalar operations as well as complex and SIMD operations and data types. CoolFlux BSP32 is fully programmable in C and comes with an optimizing compiler and tool suite.

More information about the CoolFlux BSP32 can be received from the NXP CoolFlux team (coolflux.support@nxp.com). Documentation on the tools (compiler, simulator, linker, ...) as well as documentation on the architecture and instruction set architecture are available as well. Training on programming the CoolFlux BSP32 processor can also be requested using the same channel and email address.

This module has a dedicated clock and reset signal.

21.2.1 Block diagram

The following figure shows the block diagram of the CoolFlux BSP32 subsystem.

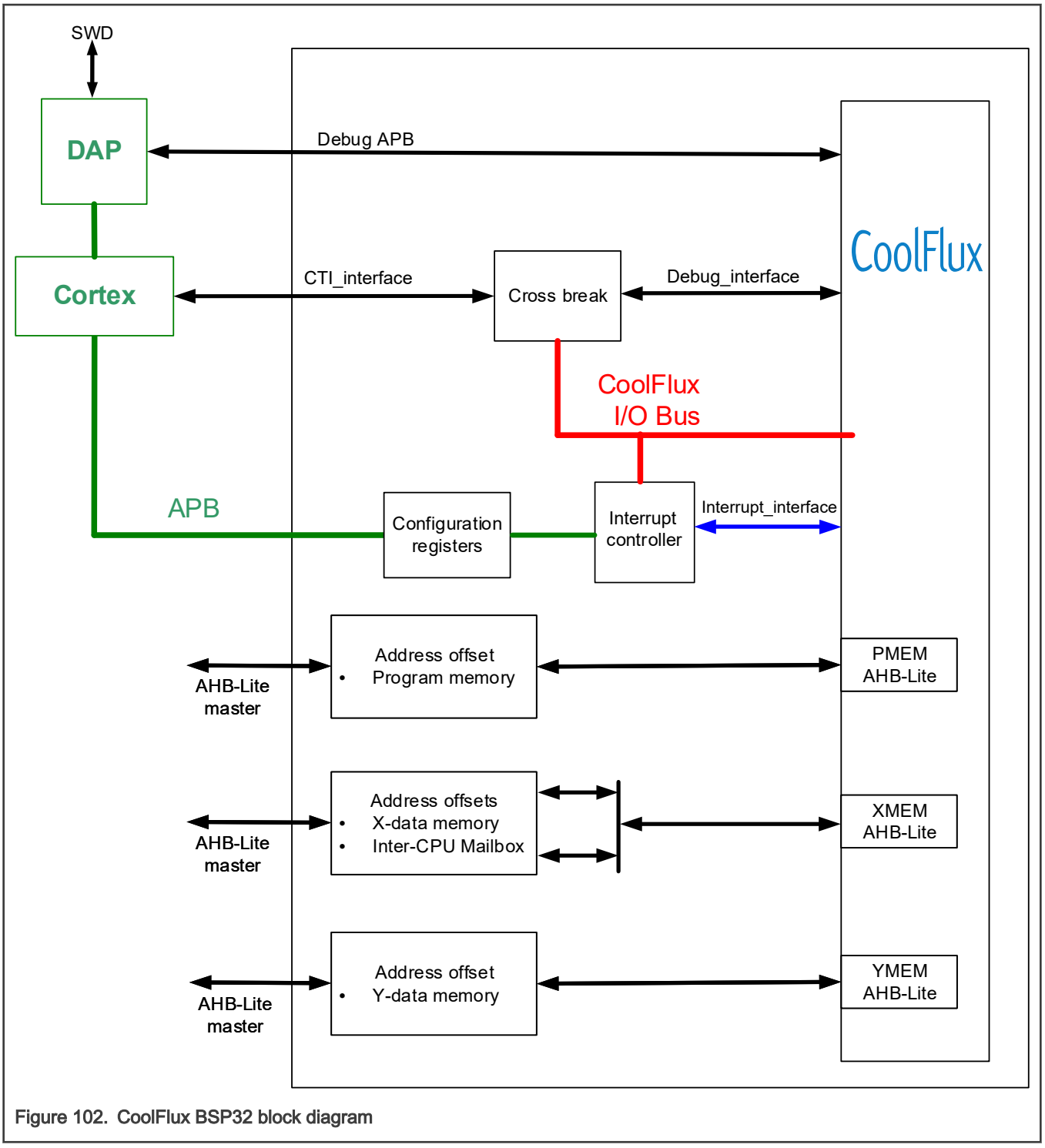


Figure 102. CoolFlux BSP32 block diagram

21.2.2 Features

- CoolFlux BSP32 32-bit processor, developed especially for audio, sensor or modem applications
- AHB-Lite interface towards program memory
- Two AHB-Lite interfaces towards on-chip data memory
- Serial Wire Debug (shared with Cortex-M33 Control Domain CPU).

21.3 Functional description

The CoolFlux BSP32 subsystem is built around the CoolFlux BSP32 processor as the main building blocks. This is a fully (C) programmable architecture that is able to execute a wide range of different algorithms. All other modules within the subsystem are meant to control this processor or to enable external communication towards the CoolFlux BSP32.

The interrupt controller can be used to interrupt the CoolFlux BSP32 based upon external events. In the other direction, the CoolFlux BSP32. also has the ability to send interrupts externally which allow inter-processor communication.

A cross-break module is available which allows common debugging between the Cortex processor and CoolFlux BSP32. This cross-break is able to transfer break events between both processors which allow a processor to (conditionally) enter debug mode at the moment the other one does.

A range of programmable registers (within the APB addressing space) is to configure and initialize the CoolFlux BSP32. This includes setting memory offset registers and inter-processor communication registers as will be explained later in the register definition chapter.

21.3.1 Security

CoolFlux BSP32 does not include any mechanism for supporting system security of the type implemented by the Cortex-M33.

21.3.2 APB programming interface

The CoolFlux BSP32 subsystem has an APB programming interface that is used to configure the subsystem. It is used to set the memory offset registers, program the internal FIFO’s, set the boot address, and generate an interrupt towards CoolFlux BSP32. Most of these registers must be set before the subsystem is pulled out of reset.

21.3.3 Cross break

The CoolFlux BSP32 subsystem contains a programmable cross break module that makes it possible to do co-debugging between CoolFlux BSP32 and Cortex M33. It can, for instance, be used to stop one processor when the other one hits a breakpoint.

21.3.4 Inter-CPU communications

CoolFlux BSP32 and CM33 can communicate with each other via various means. Shared memory that is accessible to both processors makes it possible to work on shared buffers. The mailbox can be used to send messages from one processor to the other. Finally, both processors can generate interrupts on the other one.

21.3.5 Operations and modes

The CoolFlux BSP32 is a programmable architecture that is initialized through a range of programmable registers in APB addressing space. It always executes its program via program memory and accesses the data within X or Y memory space. Hence it has no different operations or modes.

21.3.6 Clocking

The CoolFlux BSP32 has a single clock input pin called "clk". Only the rising edge of this clock is used. Note that the APB programming interface and the APB debug interface have their own APB clock signals, called "apb_pclk" and "apb_pclk_debug" respectively, both asynchronous and active low.

21.3.7 Reset

The CoolFlux BSP32 is held in reset after a device reset. Before CoolFlux BSP32 begins execution, the Cortex-M33 must load code into the memories accessible by the CoolFlux BSP32, configure CoolFlux BSP32 configuration registers (for instance, the memory offset values), and release the CoolFlux BSP32 from reset. The reset input signal is called "reset" and is asynchronous, active high. Note that the APB programming interface and the APB debug interface have their own APB reset signals, called "apb_presetn" and "apb_presetn_debug" respectively, both asynchronous and active low.

21.3.8 Interrupts

CoolFlux BSP32 has 32 external interrupt inputs. An internal interrupt controller maps these 32 interrupt sources to any of the three CoolFlux interrupt routines. Each interrupt can be enabled or disabled individually and routed to any interrupt routine. The interrupts are handled in software by the processor itself using edge detection. Whenever an interrupt is received on the CoolFlux BSP32, the corresponding interrupt routine is executed.

More explanation on the usage of interrupts can be found in the CoolFlux documentation that can be obtained from the CoolFlux team (coolflux.support@nxp.com).

The CoolFlux BSP32 subsystem can generate an external interrupt. This interrupt can be routed towards the Cortex M33 to facilitate inter-processor communication.

21.4 External signals

This section explains the external signals. The CoolFlux BSP32 subsystem has the following interface groups

External signals:

- Clock, reset and test signals
- APB programming interface
- APB debug interface
- Cross-trigger interface
- Interrupt interface
- Sleep mode indication
- Program memory AHB-Lite interface
- X-data memory AHB-Lite interface
- Y-data memory AHB-Lite interface

21.5 Initialization

Before the CoolFlux BSP32 subsystem can be used, it must be initialized. This involves the following.

Initialization:

- The three memories (P,X,Y) must be loaded with the correct program and data
- The four IVT registers (registers ivt0, ivt1, ivt2, ivt3) must be initialized
- The APB configuration registers must be initialized (e.g. offset registers to be able to access the memories)
- Clock must be enabled
- Reset must be de-asserted

From this moment onwards, the CoolFlux BSP32 will start executing the code as placed inside the program memory.

21.6 Application information

The CoolFlux BSP32 subsystem is built around the CoolFlux BSP32 processor. As this is a fully C-programmable processor, it can be used to execute different sorts of applications. These can be audio application, baseband algorithms, ultra-wide band applications. It is optimized to efficiently run integer, fix-point and complex data types in an ultra-low power manner.

21.7 Physical memory map

The general features of the CoolFlux BSP32 physical memory map are discussed in the following sections.

The CoolFlux BSP32 includes three AHB-Lite bus master interfaces. The priority of this and other AHB bus masters can be adjusted to achieve system performance needed by an application by using the SYSCON[AHBMATPRIO] register.

21.7.1 RAM

The CoolFlux BSP32 AHB-Lite program memory interface can access the SRAM X memory as well as SRAM A … SRAM H memory.

The X and Y CoolFlux BSP32 AHB-Lite data memory interfaces can access the SRAM A … SRAM H memories. These data memories also have access to the PowerQuad addressing space.

21.7.2 AHB and APB peripherals

The CoolFlux BSP32 processor access to peripherals is limited to the Inter-CPU mailbox attached to the AHB bus. This way, it is possible to communicate through short messages and interrupts between the CoolFlux BSP32 and the Cortex M33. Refer to the Inter-CPU Mailbox chapter for more information.

21.8 CoolFlux BSP32 register descriptions

Different registers to control the CoolFlux BSP32 subsystem. Before using the CoolFlux BSP32 subsystem it must be initialized using the APB interface. Communication between the CoolFlux BSP32 subsystem and the Cortex subsystem also uses this interface.

21.8.1 CoolFluxBSP32_subsystem memory map

BSP32_0 base address: 4003_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Offset address register for program memory (offset_pmem)	32	RW	0000_0000h
4h	Offset address register for X-data memory (offset_xmem)	32	RW	0000_0000h
8h	Offset address register for Y-data memory (offset_ymem)	32	RW	0000_0000h
Ch	Offset address register for mailbox peripheral (offset_mailbox)	32	RW	0000_0000h
10h	External interrupt register (interrupts_external)	32	RW	0000_0000h
14h	Interrupt status register (interrupts_status)	32	RW	0000_0000h
18h	CoolFlux BSP32 gating override (cf_gating_override)	32	RW	0000_0000h
1Ch	CoolFlux BSP32 IVT offset register (ivt_offset)	32	RW	0000_0000h
20h	CoolFlux BSP32 sleep mode register (sleep_mode)	32	R	0000_0000h
24h	CoolFlux BSP32 IVT register 0 content (ivt0)	32	RW	0000_0008h
28h	CoolFlux BSP32 IVT register 1 content (ivt1)	32	RW	0000_0009h
2Ch	CoolFlux BSP32 IVT register 2 content (ivt2)	32	RW	0000_000Ah
30h	CoolFlux BSP32 IVT register 3 content (ivt3)	32	RW	0000_000Bh
34h	CoolFlux BSP32 IVT disable register (ivt_disable)	32	RW	0000_0000h

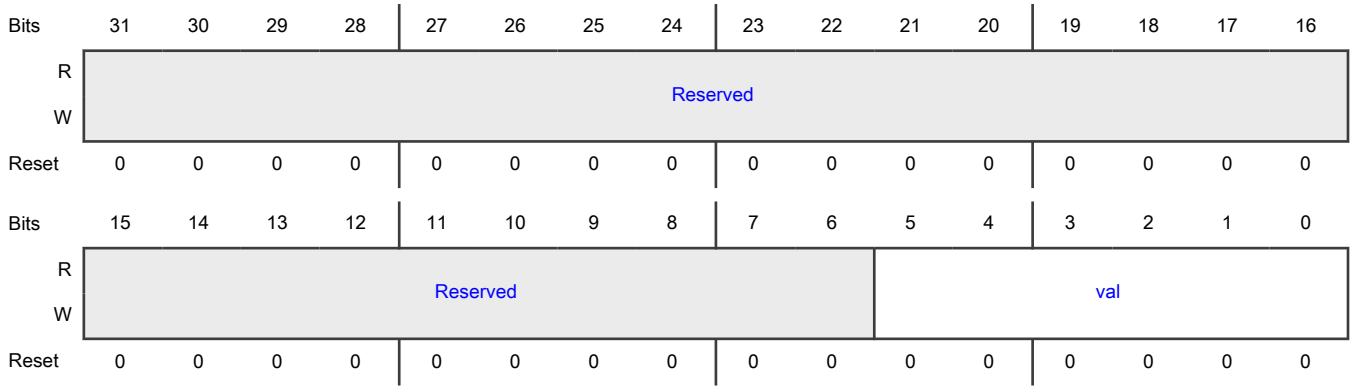
21.8.2 Offset address register for program memory (offset_pmem)

Offset

Register	Offset
offset_pmem	0h

Function
Contains the upper 6 bits to generate the AHB-Lite address for program memory.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 val	Offset address register for program memory

21.8.3 Offset address register for X-data memory (offset_xmem)

Offset

Register	Offset
offset_xmem	4h

Function
Contains the upper 8 bits to generate the AHB-Lite address for X-data memory.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								val							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 val	Offset address register for X-data memory

21.8.4 Offset address register for Y-data memory (offset_ymem)

Offset

Register	Offset
offset_ymem	8h

Function

Contains the upper 8 bits to generate the AHB-Lite address for Y-data memory.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								val							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 val	Offset address register for Y-data memory

21.8.5 Offset address register for mailbox peripheral (offset_mailbox)

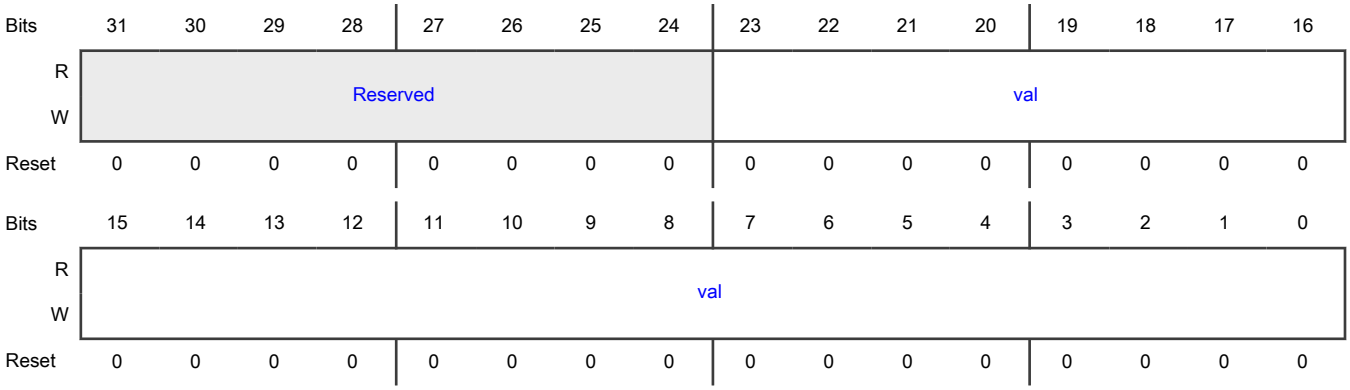
Offset

Register	Offset
offset_mailbox	Ch

Function

The upper 256 addresses (addresses 16,776,960 to 16,777,215) of the X memory interface can be used to access the Inter-CPU Mailbox peripheral. To access this peripheral, the 24-bit offset_mailbox register have to be set to 0x400B20 to generate the AHB-Lite address for the Inter-CPU Mailbox peripheral.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 val	Offset address register for mailbox peripheral

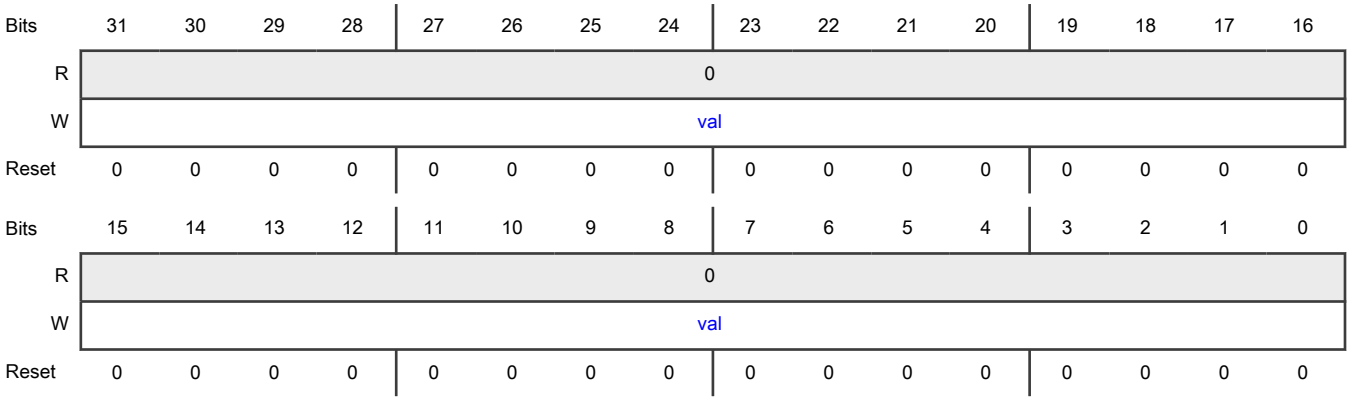
21.8.6 External interrupt register (interrupts_external)

Offset

Register	Offset
interrupts_external	10h

Function
Register to generate interrupts on the CoolFlux BSP32 from the Cortex side.

Diagram



Fields

Field	Function
31-0 val	External interrupt register

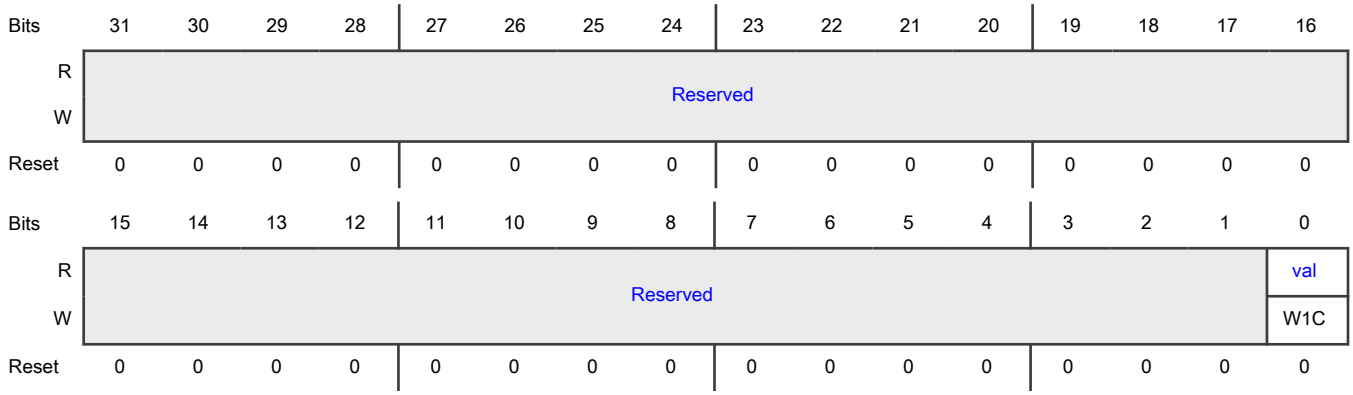
21.8.7 Interrupt status register (interrupts_status)

Offset

Register	Offset
interrupts_status	14h

Function
Register used to generate an interrupt from the CoolFlux BSP32 towards the Cortex subsystem. This register can be read and cleared (not set) by the APB interface.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 val	Interrupt status register

21.8.8 CoolFlux BSP32 gating override (cf_gating_override)

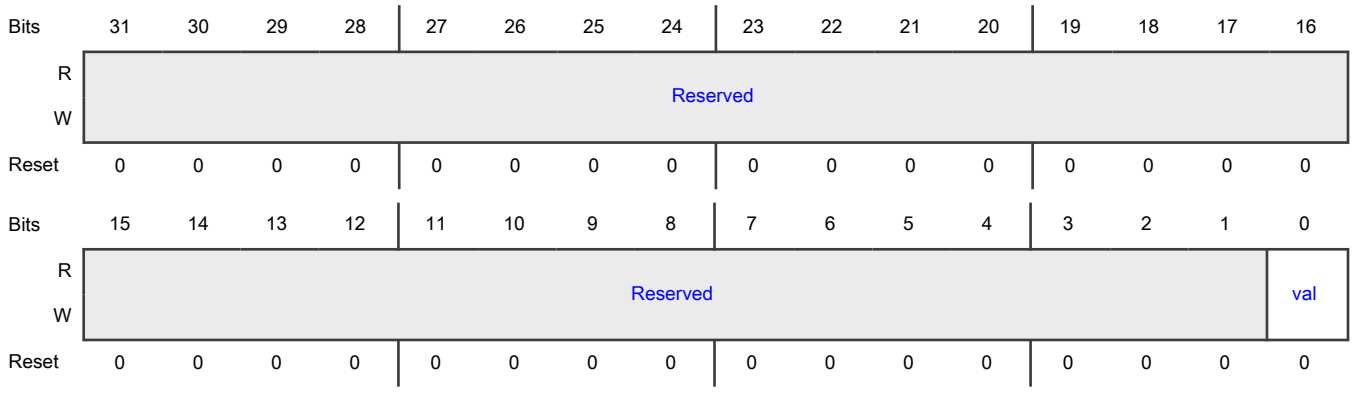
Offset

Register	Offset
cf_gating_override	18h

Function

By default, the clock towards the CoolFlux BSP32 is switched off when the processor enters sleep mode. This can be overruled by setting this bit. When set, the clock towards the CoolFlux BSP32 keeps running when it enters sleep mode. This will have a negative impact on power.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 val	CoolFlux BSP32 gating override

21.8.9 CoolFlux BSP32 IVT offset register (ivt_offset)

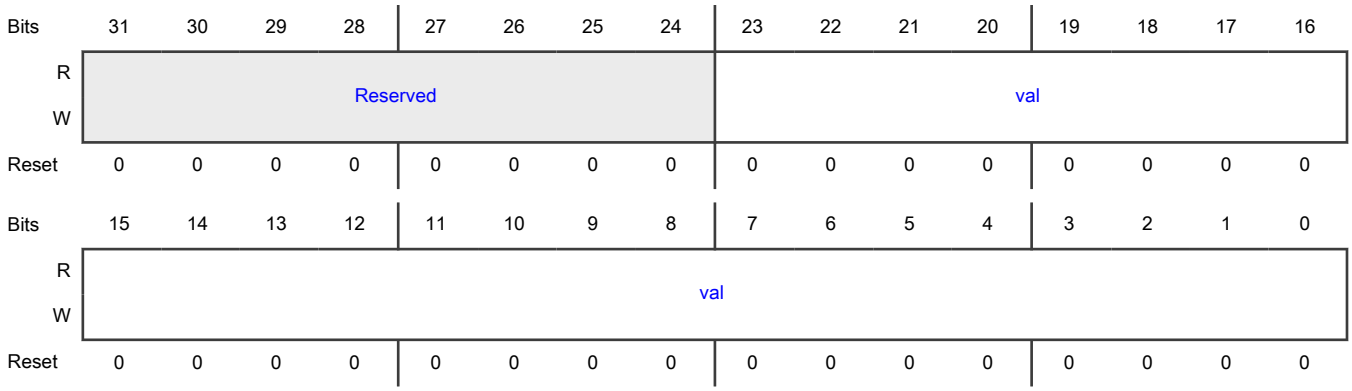
Offset

Register	Offset
ivt_offset	1Ch

Function

The Interrupt Vector Table (IVT) of the CoolFlux BSP32 is used to determine the addresses of the reset vector and interrupt vectors. By default, the IVT table starts from program memory address 0 from a CoolFlux point of view, which means that the IVT start address is defined by the "offset address for program memory" register. The IVT offset register can be used to re-locate the IVT table of the CoolFlux BSP32 program, thereby placing the IVT table in a different location in program memory. For more information about the IVT table, please refer to the CoolFlux BSP32 C Programmer's manual that can be obtained from the CoolFlux team (coolflux.support@nxp.com).

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 val	CoolFlux BSP32 IVT offset register

21.8.10 CoolFlux BSP32 sleep mode register (sleep_mode)

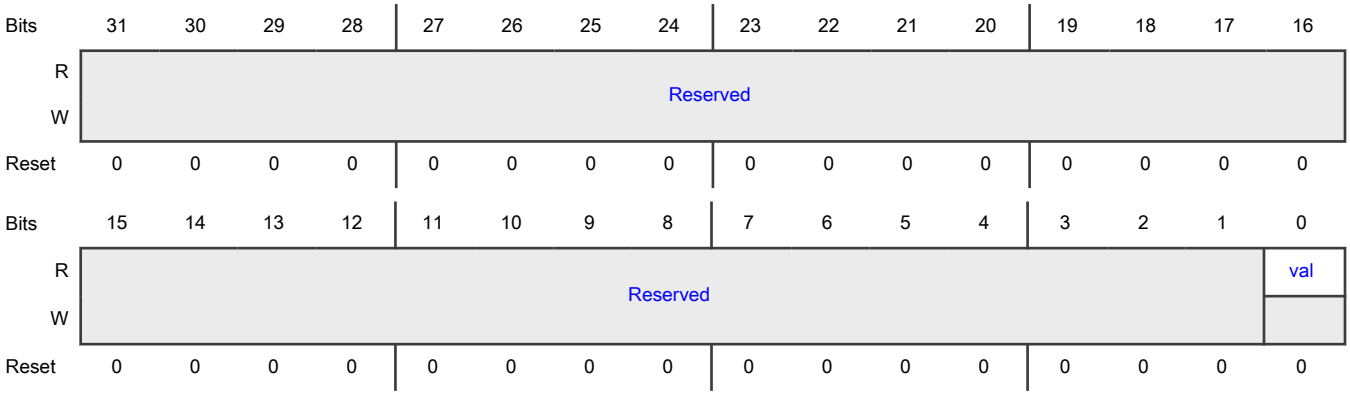
Offset

Register	Offset
sleep_mode	20h

Function

Register indicating whether the CoolFlux BSP32 is in sleep mode (1) or not (0). The CoolFlux BSP32 will enter sleep mode when the eop (End-Of-Program) instruction is executed and is thus fully under software control. It will exit from sleep mode in case of an interrupt request, debug request or a full reset. Sleep mode is used in an application to save power while waiting on a particular event (mostly an interrupt as mentioned above). In this sleep mode, power consumption is reduced as most of the clocks will be switched off (automatically done in hardware).

Diagram



Fields

Field	Function
31-1 —	Reserved
0 val	CoolFlux BSP32 sleep mode register

21.8.11 CoolFlux BSP32 IVT register 0 content (ivt0)

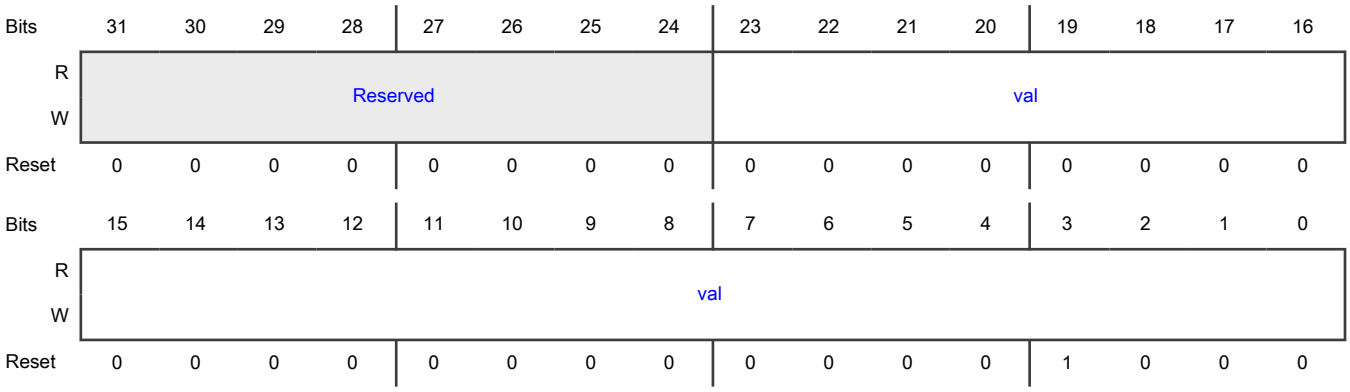
Offset

Register	Offset
ivt0	24h

Function

Content of IVT register 0 (reset vector). This needs to be written explicitly before the processor is pulled out of reset. The content of this register is used as the target address of a "goto" instruction.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 val	CoolFlux BSP32 IVT register 0 content

21.8.12 CoolFlux BSP32 IVT register 1 content (ivt1)

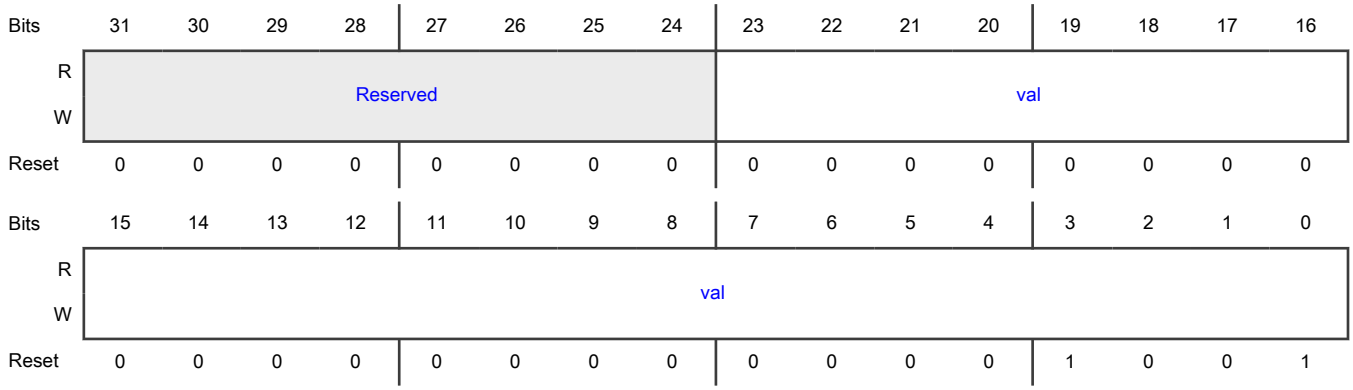
Offset

Register	Offset
ivt1	28h

Function

Content of IVT register 1 (interrupt line 0 vector). This needs to be written explicitly before the processor is pulled out of reset. The content of this register is used as the target address of a "goto" instruction.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 val	CoolFlux BSP32 IVT register 1 content

21.8.13 CoolFlux BSP32 IVT register 2 content (ivt2)

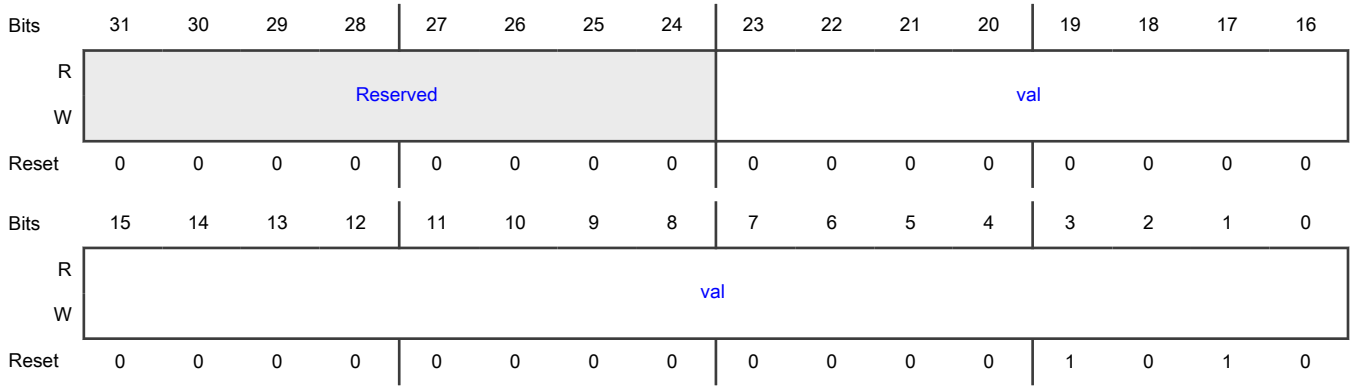
Offset

Register	Offset
ivt2	2Ch

Function

Content of IVT register 2 (interrupt line 1 vector). This needs to be written explicitly before the processor is pulled out of reset. The content of this register is used as the target address of a "goto" instruction.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 val	CoolFlux BSP32 IVT register 2 content

21.8.14 CoolFlux BSP32 IVT register 3 content (ivt3)

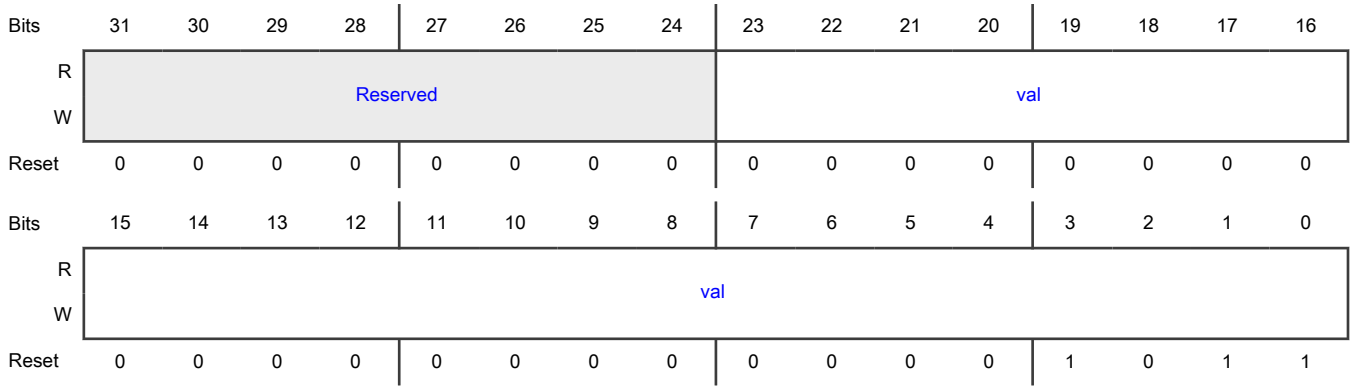
Offset

Register	Offset
ivt3	30h

Function

Content of IVT register 3 (interrupt line 2 vector). This needs to be written explicitly before the processor is pulled out of reset. The content of this register is used as the target address of a "goto" instruction.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 val	CoolFlux BSP32 IVT register 3 content

21.8.15 CoolFlux BSP32 IVT disable register (ivt_disable)

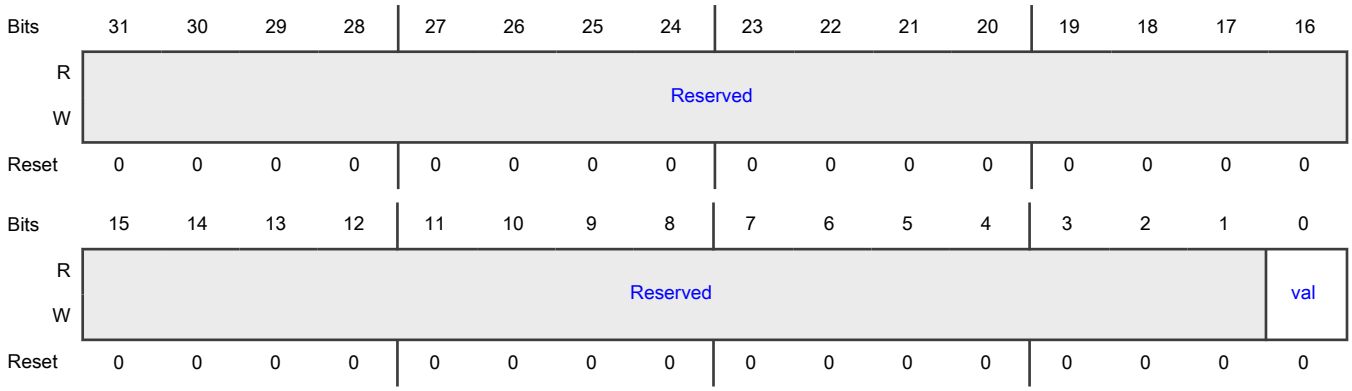
Offset

Register	Offset
ivt_disable	34h

Function

Before using the CoolFlux BSP32, the four IVT registers (registers ivt0, ivt1, ivt2, ivt3) must be initialized. These writes can be omitted by writing to this register. In that case, the four IVT registers are not used and their data is read directly from program memory instead. As this leads to longer timing paths, the maximum allowed frequency of the CoolFlux BSP32 is reduced. Its usage was deprecated in newer version and therefore, writing a "1" to this register is strongly disadvised.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 val	CoolFlux BSP32 IVT disable register

Chapter 22
Inter-CPU Mailbox (MAILBOX)

22.1 Chip-specific MAILBOX information

Table 289. Reference links to related information

Topic	Related module	Reference
Full description	MAILBOX	MAILBOX
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

This block is a AHB slave peripheral with a single clock domain and reset. This device supports two mailboxes, where each mailbox has three 32-bit registers. Each mailbox has one mutex register and one peripheral ID register. For each mailbox, the corresponding interrupt line intr[i] is set if MBOX_REG[i] is not 0 and cleared if it is 0.

22.1.1 Module instances

This device has one instance of the Inter-CPU Mailbox module, MAILBOX0.

22.2 Overview

MAILBOX provides simple communication and resource allocation between CPUs by using two mechanisms:

The first mechanism helps you send an interrupt, along with a simple message, to another CPU. To generate an interrupt to another CPU, you can set any one of the interrupt flags (up to 32) by using Cortex-M33 (CPU0) Interrupt Set (IRQ0SET) and CoolFlux (CPU1) Interrupt Set (IRQ1SET). The other CPU reads Cortex-M33 (CPU0) Interrupt Set (IRQ0SET) and CoolFlux (CPU1) Interrupt Set (IRQ1SET) to determine the action it needs to perform. If more than one IRQ*n* field is 1, the CPU receiving the interrupts reads all of them. After completing the requested service or operation, the CPU that receives the interrupt requests writes 1 to one or more fields in Cortex-M33 (CPU0) Interrupt Clear (IRQ0CLR) and CoolFlux (CPU1) Interrupt Clear (IRQ1CLR) to clear them.

To check whether the operation is complete, the CPU sending the request can check IRQ*n*, or the CPU receiving the original interrupt can use the same mechanism on the reverse to notify the requesting CPU.

You can determine the use of each bit in Cortex-M33 (CPU0) Interrupt (IRQ0) and CoolFlux (CPU1) Interrupt (IRQ1) and also provide additional information to each request bit. For example, both the CPUs can use predefined memory locations to hold detailed information about the interrupt requests.

The second mechanism is based on a single-bit resource allocation request. MUTEX[EX] defines the mutual exclusion request, and reading it provides the current status of the resource. If the resource is available, it is reserved for the process that reads the register. You define the resource control, and after getting access to the resource, you must write 1 to MUTEX[EX] to clear the field after it has used the resource. For more information, see MUTEX[EX].

22.2.1 Block diagram

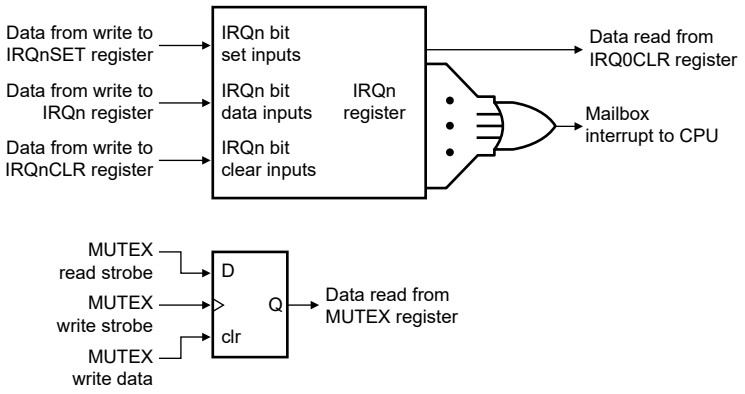


Figure 103. Block diagram

22.2.2 Features

- Provides inter-processor communication, allowing multiple CPUs to share resources and communicate with each other in a simple manner.
- Enables each CPU to generate up to 32 user-defined interrupts.
- Enables each CPU to claim a shared resource if the resource is available.

22.3 Functional description

22.3.1 Clocks

SYSCON.AHBCLKCTRL0[MAILBOX] controls this module's bus clock.

22.3.2 Reset

SYSCON.PRESETCTRL0[MAILBOX_RST] resets this module.

22.4 Signals

This module has no configurable pins.

22.5 Memory map and register definition

This section includes this module's memory map and detailed descriptions of all registers.

22.5.1 MAILBOX register descriptions

22.5.1.1 MAILBOX memory map

MAILBOX0 base address: 400B_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Cortex-M33 (CPU0) Interrupt (IRQ0)	32	RW	0000_0000h
4h	Cortex-M33 (CPU0) Interrupt Set (IRQ0SET)	32	W	See section
8h	Cortex-M33 (CPU0) Interrupt Clear (IRQ0CLR)	32	W	See section
10h	CoolFlux (CPU1) Interrupt (IRQ1)	32	RW	0000_0000h
14h	CoolFlux (CPU1) Interrupt Set (IRQ1SET)	32	W	See section
18h	CoolFlux (CPU1) Interrupt Clear (IRQ1CLR)	32	W	See section
F8h	Mutual Exclusion (MUTEX)	32	RW	0000_0001h

22.5.1.2 Cortex-M33 (CPU0) Interrupt (IRQ0)

Offset

Register	Offset
IRQ0	0h

Function

Allows other CPUs to send interrupt requests to Cortex-M33 (CPU0).

This register allows communication between CPUs. For example, a CPU can handle certain peripherals, along with signaling another CPU, when data is available. Each bit of this register can be configured differently, based on your requirements.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTREQ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTREQ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 INTREQ	Interrupt Request Sends interrupt requests to the Cortex-M33 (CPU0) interrupt controller, if you write 1 to any of the bits.

22.5.1.3 Cortex-M33 (CPU0) Interrupt Set (IRQ0SET)

Offset

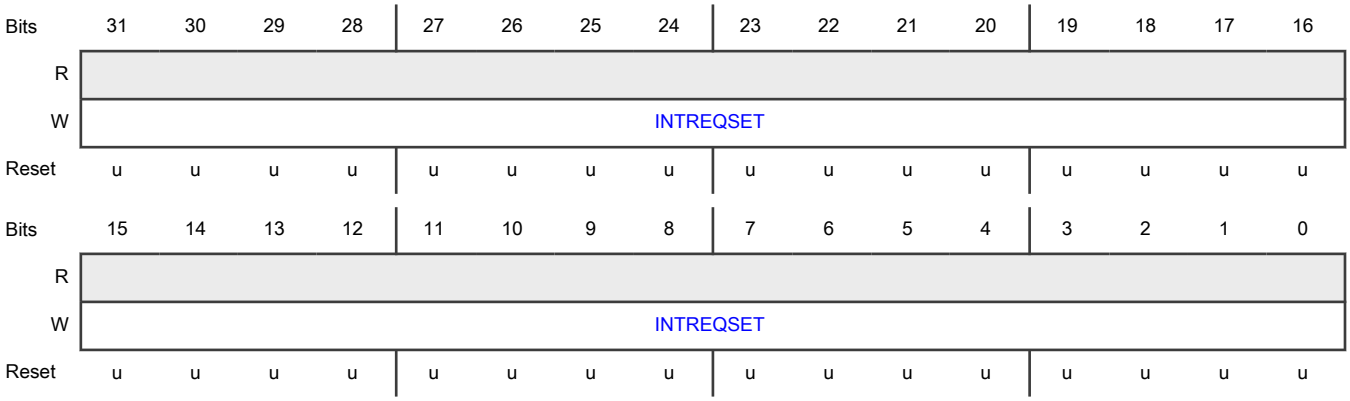
Register	Offset
IRQ0SET	4h

Function

Generates interrupt requests for Cortex-M33 (CPU0).

Writing 1 to any bit of this register writes 1 to the corresponding bit of Cortex-M33 (CPU0) Interrupt (IRQ0). You can generate more than one interrupt by writing 1 to multiple bits of this register.

Diagram



Fields

Field	Function
31-0 INTREQSET	Interrupt Request Set 0 Generates interrupt requests for Cortex-M33 (CPU0).

22.5.1.4 Cortex-M33 (CPU0) Interrupt Clear (IRQ0CLR)

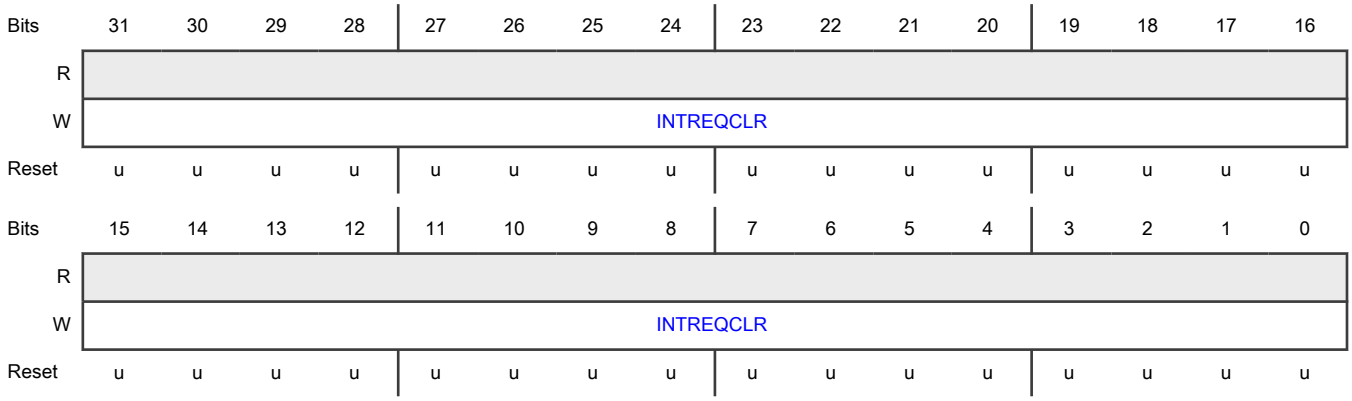
Offset

Register	Offset
IRQ0CLR	8h

Function

Clears interrupts for Cortex-M33 (CPU0) after the requested service or operation is complete. Writing 1 to any bit of this register writes 0 to the corresponding bit of Cortex-M33 (CPU0) Interrupt (IRQ0).

Diagram



Fields

Field	Function
31-0 INTREQCLR	Interrupt Request Clear 0 Clears interrupt requests for Cortex-M33 (CPU0).

22.5.1.5 CoolFlux (CPU1) Interrupt (IRQ1)

Offset

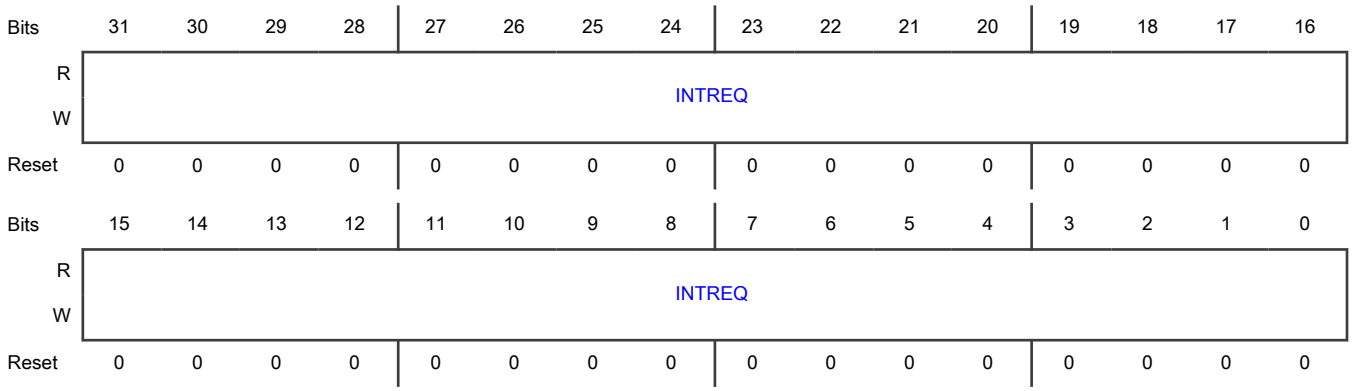
Register	Offset
IRQ1	10h

Function

Allows other CPUs to send interrupt requests to CoolFlux (CPU1).

This register allows communication between CPUs. For example, a CPU can handle certain peripherals, along with signaling another CPU, when data is available. Each bit of this register can be configured differently, based on your requirements.

Diagram



Fields

Field	Function
31-0 INTREQ	Interrupt Request Sends interrupt requests to the CoolFlux (CPU1) interrupt controller, if you write 1 to any of the bits.

22.5.1.6 CoolFlux (CPU1) Interrupt Set (IRQ1SET)

Offset

Register	Offset
IRQ1SET	14h

Function

Generates interrupt requests for CoolFlux (CPU1).

Writing 1 to any bit of this register writes 1 to the corresponding bit of CoolFlux (CPU1) Interrupt (IRQ1). You can generate more than one interrupt by writing 1 to multiple bits of this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	INTREQSET															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	INTREQSET															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 INTREQSET	Interrupt Request Set 1 Generates interrupt requests for CoolFlux (CPU1).

22.5.1.7 CoolFlux (CPU1) Interrupt Clear (IRQ1CLR)

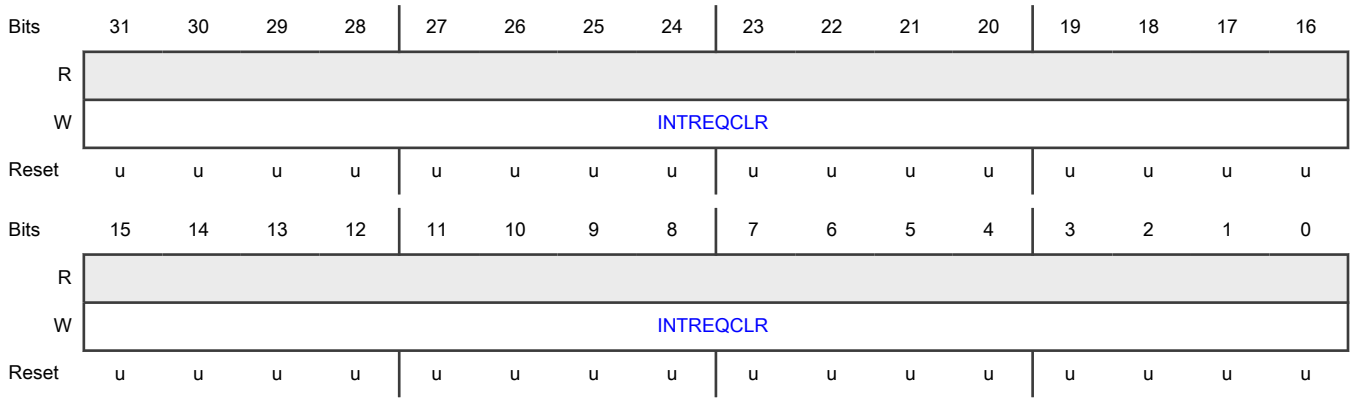
Offset

Register	Offset
IRQ1CLR	18h

Function

Clears interrupts for CoolFlux (CPU1) after the requested service or operation is complete. Writing 1 to any bit of this register writes 0 to the corresponding bit of CoolFlux (CPU1) Interrupt (IRQ1).

Diagram



Fields

Field	Function
31-0 INTREQCLR	Interrupt Request Clear 1 Clears interrupt requests for CoolFlux (CPU1).

22.5.1.8 Mutual Exclusion (MUTEX)

Offset

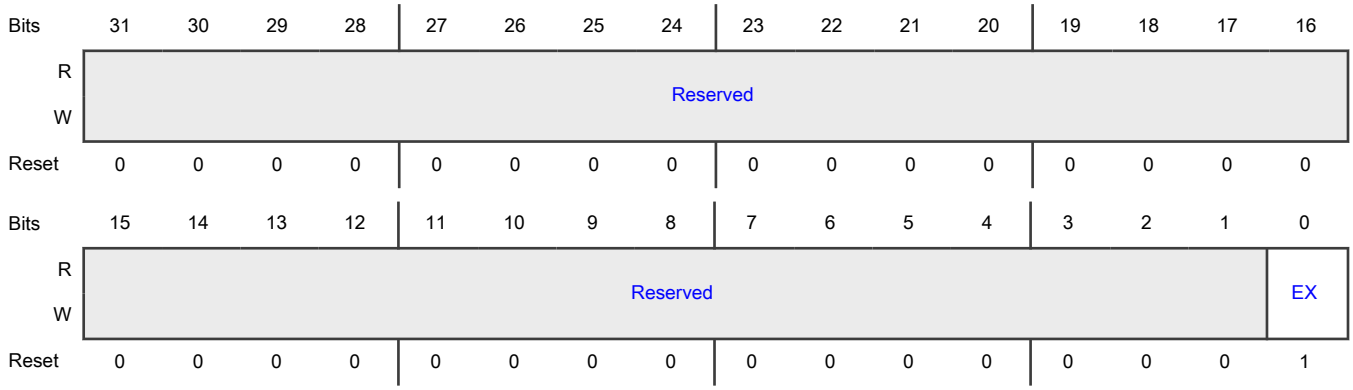
Register	Offset
MUTEX	F8h

Function

Provides an inter-processor communication handshake.

This register provides a resource allocation handshake between two CPUs. To access a shared resource (possibly a resource allocation table in memory), a CPU reads this register. If the CPU reads 1, it has control over the shared resource allocation. After making the required changes, the CPU writes to the register. As a result, MUTEX[EX] becomes 1 again, making the shared resource allocation available to another CPU. If the CPU reads 0, it must wait for MUTEX[EX] to become 1 before accessing the shared resource allocation information.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 EX	Mutual Exclusion Request Provides resource allocation handshake between two CPUs. This field indicates the availability of a shared resource. When read for any reason, the field returns the current value and becomes 0. The field becomes 1 again following any write. 0b - Resource unavailable 1b - Resource available

Chapter 23
Enhanced Direct Memory Access (eDMA) Controller

23.1 Chip-specific eDMA information

Table 290. Reference links to related information

Topic	Related module	Reference
Full description	eDMA	eDMA
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See DMAn_REQ_ENABLEn registers in INPUTMUX

23.1.1 Module instances

This device has two 16 channel eDMA controllers.

23.1.2 Direct Memory Access Multiplexer (DMAMUX)

The peripheral request line for each channel of the Direct Memory Access Controller (DMA) is driven from a Direct Memory Access Multiplexer (DMAMUX). This is a flexible configuration that allows the user to select the appropriate peripheral to connect to each channel of the DMA Controller. The DMAMUX for this device allows up to 128 DMA request signals (6 unused signals are reserved) to be mapped to each channel. Because of the mux, there is not a correlation between any of the DMA request sources and a specific DMA channel.

The DMAMUX is an integrated component of the DMA Controller. This allows each DMA channel and DMA Multiplexer to configured as secure or non-secure on initial configuration of the device and not be changed without the correct permissions.

23.1.2.1 DMAMUX0 Request Assignments

Table 291. DMAMUX0 request assignments

DMAMUX Number	Alias	Source Description
Modules		
0	—	Disabled
1	FlexSPI0	Receive event
2	FlexSPI0	Transmit event
3	PINT0	INT0
4	PINT0	INT1
5	PINT0	INT2
6	PINT0	INT3
7	CTIMER0	DMAREQ_M0
8	CTIMER0	DMAREQ_M1

Table continues on the next page...

Table 291. DMAMUX0 request assignments (continued)

DMAMUX Number	Alias	Source Description
9	CTIMER1	DMAREQ_M0
10	CTIMER1	DMAREQ_M1
11	CTIMER2	DMAREQ_M0
12	CTIMER2	DMAREQ_M1
13	CTIMER3	DMAREQ_M0
14	CTIMER3	DMAREQ_M1
15	CTIMER4	DMAREQ_M0
16	CTIMER4	DMAREQ_M1
17	WUU0	Wake up event
18	MICFIL0	FIFO_request
19	SCT0	DMA0
20	SCT0	DMA1
21	ADC0	FIFO A request
22	ADC0	FIFO B request
23	ADC1	FIFO A request
24	ADC1	FIFO B request
25	DAC0	FIFO_request
26	DAC1	FIFO_request
27	DAC2	FIFO_request
28	CMP0	DMA_request
29	CMP1	DMA_request
30	CMP2	DMA_request
31	EVTG0	OUT0A
32	EVTG0	OUT0B
33	EVTG0	OUT1A
34	EVTG0	OUT1B
35	EVTG0	OUT2A
36	EVTG0	OUT2B
37	EVTG0	OUT3A
38	EVTG0	OUT3B
39	PWM0	Req_capt0
40	PWM0	Req_capt1

Table continues on the next page...

Table 291. DMAMUX0 request assignments (continued)

DMAMUX Number	Alias	Source Description
41	PWM0	Req_capt2
42	PWM0	Req_capt3
43	PWM0	Req_val0
44	PWM0	Req_val1
45	PWM0	Req_val2
46	PWM0	Req_val3
47	PWM1	Req_capt0
48	PWM1	Req_capt1
49	PWM1	Req_capt2
50	PWM1	Req_capt3
51	PWM1	Req_val0
52	PWM1	Req_val1
53	PWM1	Req_val2
54	PWM1	Req_val3
55	Reserved	—
56	Reserved	—
57	LPTMR0	Counter match event
58	LPTMR1	Counter match event
59	CAN0	DMA request
60	CAN1	DMA request
61	FlexIO0	Shifter0 Status DMA request OR Timer0 Status DMA request
62	FlexIO0	Shifter1 Status DMA request OR Timer1 Status DMA request
63	FlexIO0	Shifter2 Status DMA request OR Timer2 Status DMA request
64	FlexIO0	Shifter3 Status DMA request OR Timer3 Status DMA request
65	FlexIO0	Shifter4 Status DMA request OR Timer4 Status DMA request
66	FlexIO0	Shifter5 Status DMA request OR Timer5 Status DMA request
67	FlexIO0	Shifter6 Status DMA request OR Timer6 Status DMA request
68	FlexIO0	Shifter7 Status DMA request OR Timer7 Status DMA request
69	LP_FLEXCOMM0	Receive request
70	LP_FLEXCOMM0	Transmit request
71	LP_FLEXCOMM1	Receive request
72	LP_FLEXCOMM1	Transmit request

Table continues on the next page...

Table 291. DMAMUX0 request assignments (continued)

DMAMUX Number	Alias	Source Description
73	LP_FLEXCOMM2	Receive request
74	LP_FLEXCOMM2	Transmit request
75	LP_FLEXCOMM3	Receive request
76	LP_FLEXCOMM3	Transmit request
77	LP_FLEXCOMM4	Receive request
78	LP_FLEXCOMM4	Transmit request
79	LP_FLEXCOMM5	Receive request
80	LP_FLEXCOMM5	Transmit request
81	LP_FLEXCOMM6	Receive request
82	LP_FLEXCOMM6	Transmit request
83	LP_FLEXCOMM7	Receive request
84	LP_FLEXCOMM7	Transmit request
85	LP_FLEXCOMM8	Receive request
86	LP_FLEXCOMM8	Transmit request
87	LP_FLEXCOMM9	Receive request
88	LP_FLEXCOMM9	Transmit request
89	Reserved	—
90	Reserved	—
91	EMVSIM0	Receive request
92	EMVSIM0	Transmit request
93	EMVSIM1	Receive request
94	EMVSIM1	Transmit request
95	I3C0	Receive request
96	I3C0	Transmit request
97	I3C1	Receive request
98	I3C1	Transmit request
99	SAI0	Receive request
100	SAI0	Transmit request
101	SAI1	Receive request
102	SAI1	Transmit request
103	SINC0	ipd_req_sinc[0] or ipd_req_alt [0]
104	SINC0	ipd_req_sinc[1] or ipd_req_alt [1]

Table continues on the next page...

Table 291. DMAMUX0 request assignments (continued)

DMAMUX Number	Alias	Source Description
105	SINC0	ipd_req_sinc[2] or ipd_req_alt [2]
106	SINC0	ipd_req_sinc[3] or ipd_req_alt [3]
107	SINC0	ipd_req_sinc[4] or ipd_req_alt [4]
108	GPIO0	Pin event request 0
109	GPIO0	Pin event request 1
110	GPIO1	Pin event request 0
111	GPIO1	Pin event request 1
112	GPIO2	Pin event request 0
113	GPIO2	Pin event request 1
114	GPIO3	Pin event request 0
115	GPIO3	Pin event request 1
116	GPIO4	Pin event request 0
117	GPIO4	Pin event request 1
118	GPIO5	Pin event request 0
119	GPIO5	Pin event request 1
120	TSI0	End of Scan
121	TSI0	Out of Range

23.1.2.2 DMAMUX1 Request Assignments

Table 292. DMAMUX1 request assignments

DMAMUX Number	Alias	Source Description
Modules		
0	—	Disabled
1	FlexSPI0	Receive event
2	FlexSPI0	Transmit event
3	PINT0	INT0
4	PINT0	INT1
5	PINT0	INT2
6	PINT0	INT3
7	CTIMER0	DMAREQ_M0
8	CTIMER0	DMAREQ_M1

Table continues on the next page...

Table 292. DMAMUX1 request assignments (continued)

DMAMUX Number	Alias	Source Description
9	CTIMER1	DMAREQ_M0
10	CTIMER1	DMAREQ_M1
11	CTIMER2	DMAREQ_M0
12	CTIMER2	DMAREQ_M1
13	CTIMER3	DMAREQ_M0
14	CTIMER3	DMAREQ_M1
15	CTIMER4	DMAREQ_M0
16	CTIMER4	DMAREQ_M1
17	WUU0	Wake up event
18	MICFIL0	FIFO_request
19	SCT0	DMA0
20	SCT0	DMA1
21	ADC0	FIFO A request
22	ADC0	FIFO B request
23	ADC1	FIFO A request
24	ADC1	FIFO B request
25	DAC0	FIFO_request
26	DAC1	FIFO_request
27	DAC2	FIFO_request
28	CMP0	DMA_request
29	CMP1	DMA_request
30	CMP2	DMA_request
31	EVTG0	OUT0A
32	EVTG0	OUT0B
33	EVTG0	OUT1A
34	EVTG0	OUT1B
35	EVTG0	OUT2A
36	EVTG0	OUT2B
37	EVTG0	OUT3A
38	EVTG0	OUT3B
39	PWM0	Req_capt0
40	PWM0	Req_capt1

Table continues on the next page...

Table 292. DMAMUX1 request assignments (continued)

DMAMUX Number	Alias	Source Description
41	PWM0	Req_capt2
42	PWM0	Req_capt3
43	PWM0	Req_val0
44	PWM0	Req_val1
45	PWM0	Req_val2
46	PWM0	Req_val3
47	PWM1	Req_capt0
48	PWM1	Req_capt1
49	PWM1	Req_capt2
50	PWM1	Req_capt3
51	PWM1	Req_val0
52	PWM1	Req_val1
53	PWM1	Req_val2
54	PWM1	Req_val3
55	Reserved	—
56	Reserved	—
57	LPTMR0	Counter match event
58	LPTMR1	Counter match event
59	CAN0	DMA request
60	CAN1	DMA request
61	FlexIO0	Shifter0 Status DMA request OR Timer0 Status DMA request
62	FlexIO0	Shifter1 Status DMA request OR Timer1 Status DMA request
63	FlexIO0	Shifter2 Status DMA request OR Timer2 Status DMA request
64	FlexIO0	Shifter3 Status DMA request OR Timer3 Status DMA request
65	FlexIO0	Shifter4 Status DMA request OR Timer4 Status DMA request
66	FlexIO0	Shifter5 Status DMA request OR Timer5 Status DMA request
67	FlexIO0	Shifter6 Status DMA request OR Timer6 Status DMA request
68	FlexIO0	Shifter7 Status DMA request OR Timer7 Status DMA request
69	LP_FLEXCOMM0	Receive request
70	LP_FLEXCOMM0	Transmit request
71	LP_FLEXCOMM1	Receive request
72	LP_FLEXCOMM1	Transmit request

Table continues on the next page...

Table 292. DMAMUX1 request assignments (continued)

DMAMUX Number	Alias	Source Description
73	LP_FLEXCOMM2	Receive request
74	LP_FLEXCOMM2	Transmit request
75	LP_FLEXCOMM3	Receive request
76	LP_FLEXCOMM3	Transmit request
77	LP_FLEXCOMM4	Receive request
78	LP_FLEXCOMM4	Transmit request
79	LP_FLEXCOMM5	Receive request
80	LP_FLEXCOMM5	Transmit request
81	LP_FLEXCOMM6	Receive request
82	LP_FLEXCOMM6	Transmit request
83	LP_FLEXCOMM7	Receive request
84	LP_FLEXCOMM7	Transmit request
85	LP_FLEXCOMM8	Receive request
86	LP_FLEXCOMM8	Transmit request
87	LP_FLEXCOMM9	Receive request
88	LP_FLEXCOMM9	Transmit request
89	Reserved	—
90	Reserved	—
91	EMVSIM0	Receive request
92	EMVSIM0	Transmit request
93	EMVSIM1	Receive request
94	EMVSIM1	Transmit request
95	I3C0	Receive request
96	I3C0	Transmit request
97	I3C1	Receive request
98	I3C1	Transmit request
99	SAI0	Receive request
100	SAI0	Transmit request
101	SAI1	Receive request
102	SAI1	Transmit request
103	SINC0	ipd_req_sinc[0] or ipd_req_alt [0]
104	SINC0	ipd_req_sinc[1] or ipd_req_alt [1]

Table continues on the next page...

Table 292. DMAMUX1 request assignments (continued)

DMAMUX Number	Alias	Source Description
105	SINC0	ipd_req_sinc[2] or ipd_req_alt [2]
106	SINC0	ipd_req_sinc[3] or ipd_req_alt [3]
107	SINC0	ipd_req_sinc[4] or ipd_req_alt [4]
108	GPIO0	Pin event request 0
109	GPIO0	Pin event request 1
110	GPIO1	Pin event request 0
111	GPIO1	Pin event request 1
112	GPIO2	Pin event request 0
113	GPIO2	Pin event request 1
114	GPIO3	Pin event request 0
115	GPIO3	Pin event request 1
116	GPIO4	Pin event request 0
117	GPIO4	Pin event request 1
118	GPIO5	Pin event request 0
119	GPIO5	Pin event request 1
120	TSI0	End of Scan
121	TSI0	Out of Range

23.1.3 DSPI

DSPI is external to the DMA and is only referred as an example of DMA hardware request usage.

23.1.4 Security considerations

When TrustZone-M is used, the following configurations are recommended for eDMA usage:

- Use strict checking (AHBSC.MISC_CTRL_REG[DISABLE_STRICT] = 01b). Strict mode enabled is also the default configuration.
- Configure both eDMAs for secure and privileged level in the AHBSC's MASTER_SEC_LEVEL and MASTER_SEC_ANTI_POL_REG registers (**note**: this is not the default setting).
- Enable the eDMA master ID replication feature for all DMA channels. The master ID replication feature is enabled globally by setting MP_CSR[GMRC], and then is enabled on a per channel basis by setting CHn_SBR[EMI].
- AHBSC peripheral slots and CHn_SBR[PAL, SEC] for each DMA channel need to be preconfigured for secure/non-secure and privileged/non-privileged based on the intended usage of each channel. This allows masters of each type to configure allocated channels. Unused DMA channels should be configured for secure and privileged access to prevent misuse.

When the settings above are followed, the master ID is captured when the TCDn_CSR is written according to the configuration of the master that wrote TCDn_CSR. The final DMA channel access security level can be downgraded from secure to non-secure depending on the source and destination addresses. See the "Master Security Wrapper (MSW)" section of the device Security Reference Manual.

NOTE

DMA scatter/gather operations must use an address matching the configuration of the DMA channel. Secure channels must access a secure memory address to load new TCDs. Non-secure channels must access a non-secure memory address to load new TCDs.

23.2 Overview

The enhanced direct memory access (eDMA) controller is capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 16 channels

23.2.1 Block diagram

Figure 104 illustrates the components of the eDMA system, including the eDMA module (engine).

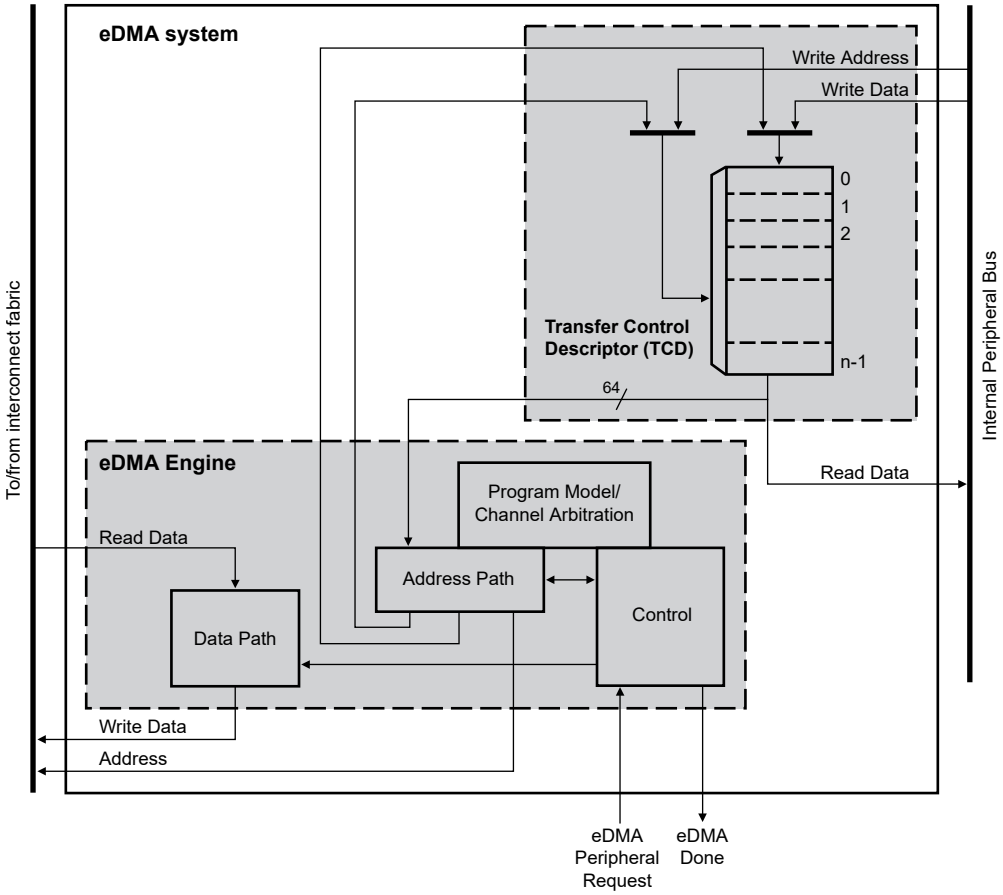


Figure 104. Block diagram

23.2.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 293. eDMA engine submodules

Submodule	Function
Address path	This block: • Implements a primary channel and secondary (preempt) channel • Manages all master bus-address calculations All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the primary channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by CH*n*_PRI[ECP]) where a large data transfer can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of a new TCD*n* from memory as part of a scatter/gather operation. See Dynamic scatter/gather for more details.
Data path	This block implements the bus master read/write data path. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase), and the data path module implements the second stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has been moved from the source to the destination. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, the eDMA performs two reads, then one 32-bit write.

The transfer control descriptor local memory is further partitioned into:

Table 294. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, and manages accesses from the eDMA engine as well as references from the internal peripheral bus. In simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

23.2.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and is not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for complex address calculations
- 16-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage for all transfers
  - Connections to the crossbar switch for bus mastering the data movement
- TCD organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel that are logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures

In the discussion of this module, *n* is used to reference the channel number.

23.3 Functional description

The operation of eDMA is described in the following subsections.

23.3.1 Modes of operation

The eDMA operates in the following modes:

Table 295. Modes of operation

Mode	Description
Normal	In Normal mode, eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with eDMA. A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the TCD. The minor loop is the sequence of read-write operations that transfers these NBYTES per service

Table continues on the next page...

Table 295. Modes of operation (continued)

Mode	Description
	request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	eDMA operation is configurable in Debug mode via the control register: • If CSR[EDBG] is cleared to 0, eDMA continues to operate. • If CSR[EDBG] is set to 1, eDMA stops transferring data. If Debug mode is entered when a channel is active, eDMA continues operation until the channel retires.

23.3.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

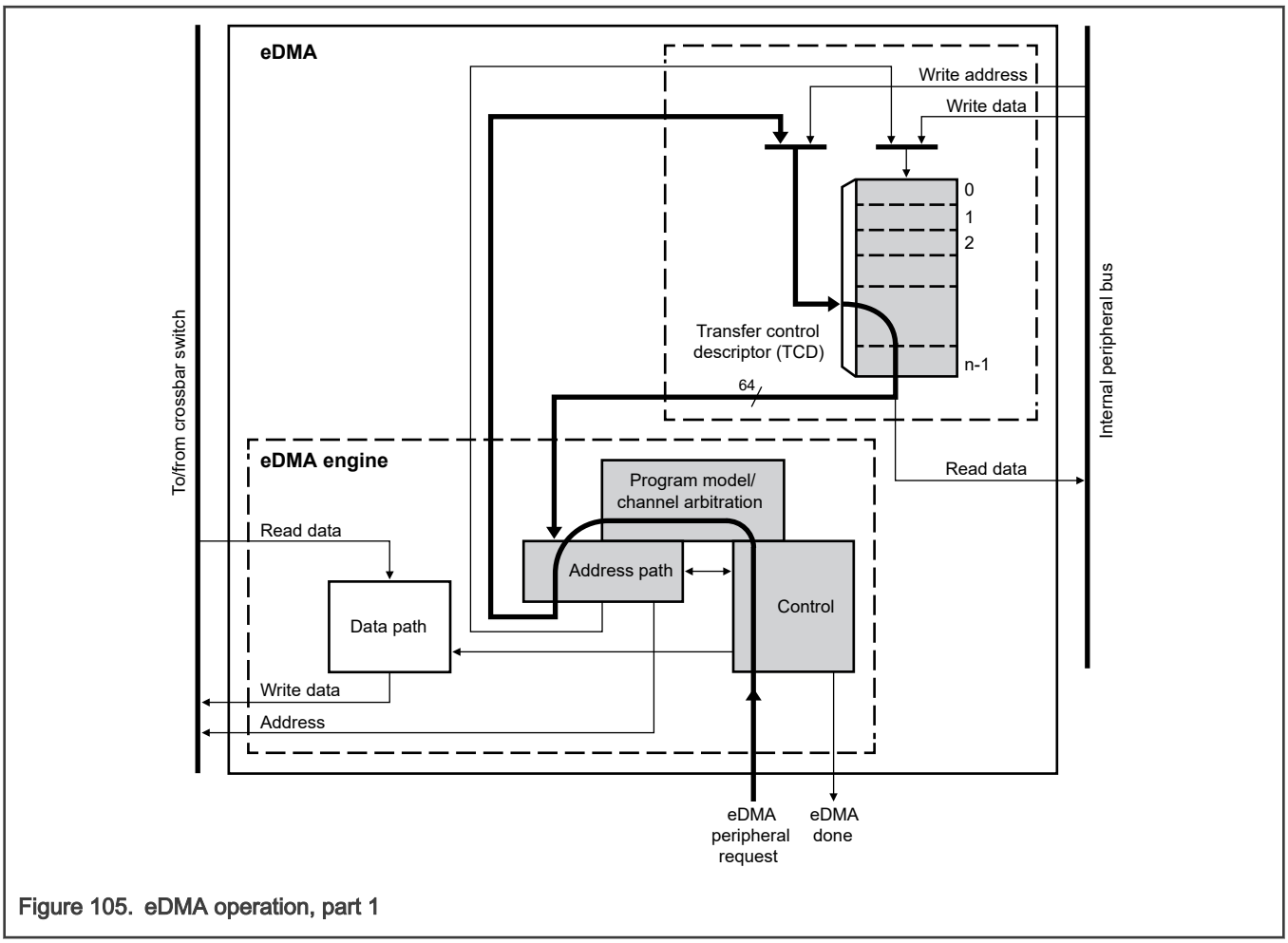


Figure 105. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] field follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration.

In the next cycle, the channel arbitration begins using fixed-priority plus the optional round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the

local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor is read from the local memory and then loaded into the eDMA engine address path's primary or secondary channel execution registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path registers.

The following diagram illustrates the second part of the basic data flow:

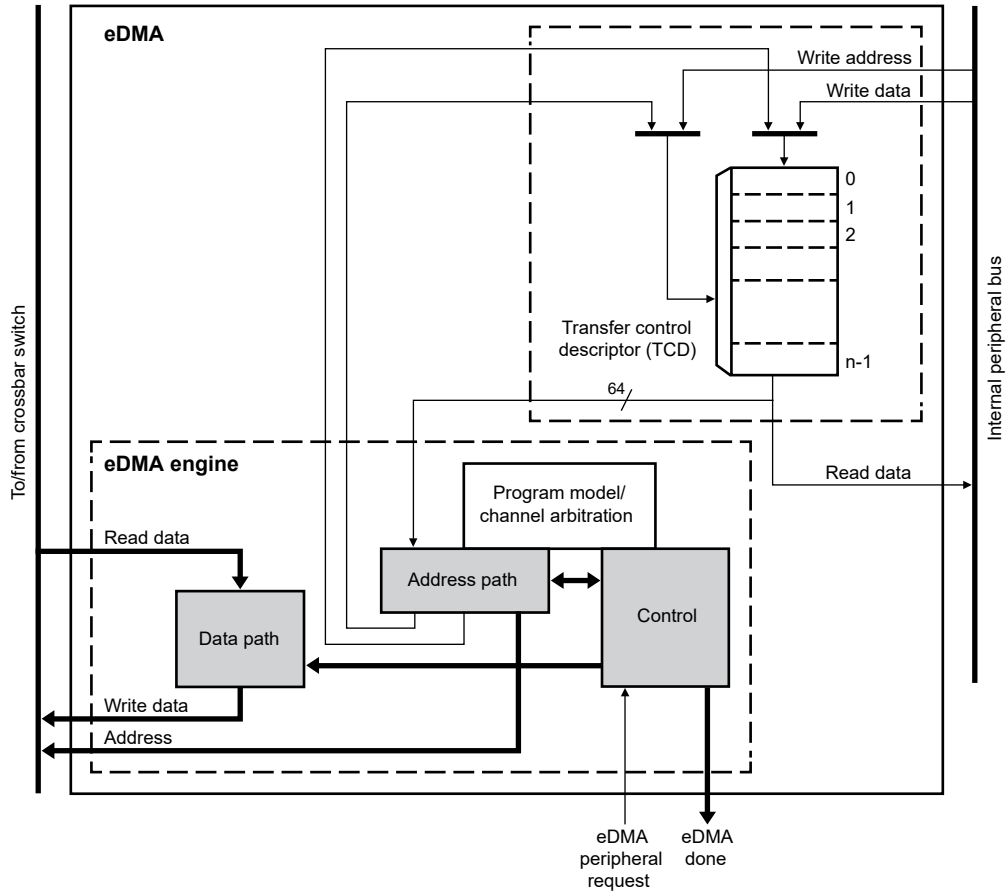


Figure 106. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) go through the required sequence of source reads and destination writes to perform the actual data movement. The source reads are initiated, and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the byte count, NBYTES, has been transferred.

After NBYTES of data has been moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD (for example, SADDR, DADDR, CITER). If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER field. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

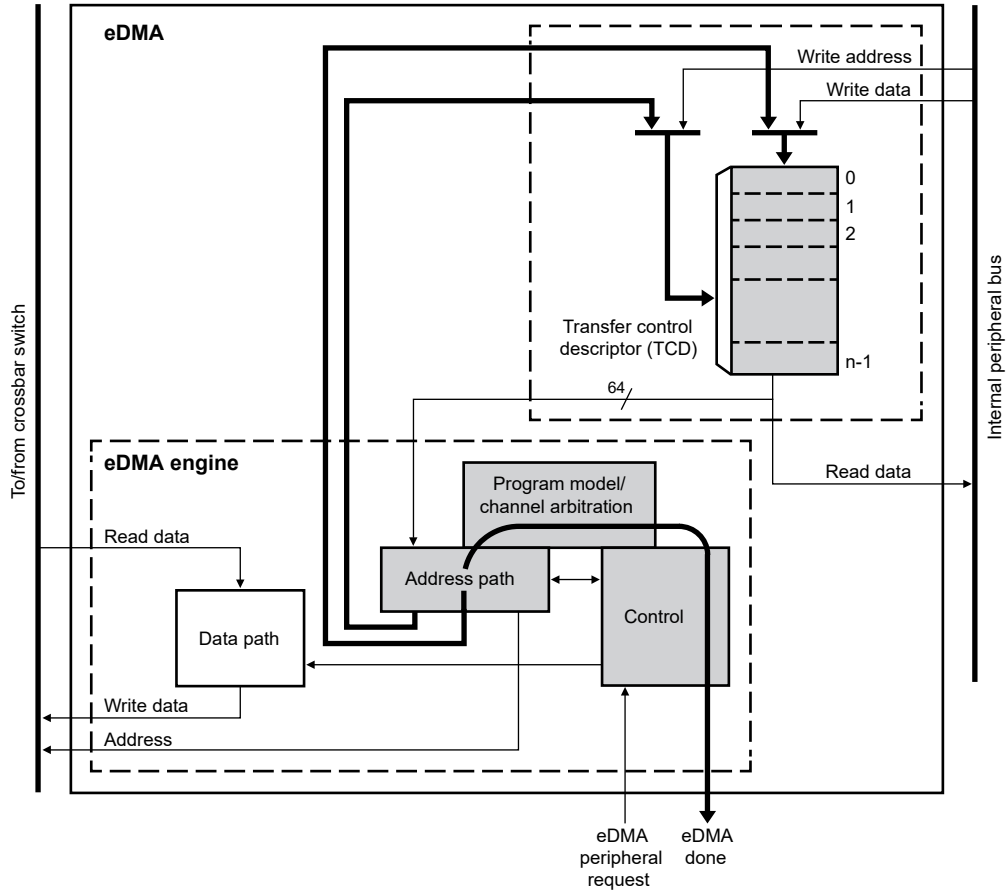


Figure 107. eDMA operation, part 3

23.3.3 Fault reporting and handling

Channel errors are reported in the Error Status register (CHn_ES) and can be caused by any of the following:

- A configuration error, which is an illegal setting in the transfer control descriptor
- An active channel canceled via a "cancel transfer with error" hardware or software request
- An error termination to a bus master read or write cycle

A configuration error is reported when an inconsistent state is represented by one of these factors:

- Starting source or destination address
- Source or destination offsets
- Minor loop byte count
- Transfer size

Each of these possible causes is detailed below:

- The addresses and offsets must be aligned on zero-modulo-transfer-sized boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size.

NOTE

To aid in debugging, set the Halt After Error field in the DMA's Control Status register, CSR[HAE]. Upon any error condition, the DMA is halted after the error is recorded. The DMA remains halted and does not process any channel service requests. After the error is fixed, the DMA may be enabled again by clearing the Halt field, CSR[HALT].

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (TCDn_DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[ELINK] field does not equal the TCDn_BITER[ELINK] field.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, are reported as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion if properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel field in the eDMA error register is set to 1. At the same time, the details of the error condition are loaded into the Error Status register (CHn_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag, and the possible assertion of an interrupt request are not affected when an error is detected.

After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

The error status fields are read-only. These error indicators are sticky and cannot be cleared. They show the last recorded error until the DMA is reset. CHn_ES[ERR] is used to determine if a new error condition exists. This field is the logical OR of each channel's error interrupt field (ERR).

After the software has resolved all errors and cleared all of the error interrupt fields, the MP_ES[VLD] is cleared to 0 but the cause of the last error is still indicated.

23.3.4 Channel preemption

The eDMA uses a priority vector value to determine the highest priority channel requesting service.

The priority vector is a combination of:

1. the channel's group priority, CHn_GRPRI
2. the channel's priority level, CHn_PRI[APL]

It can be considered a number composed of these concatenated priority levels: CHn_GRPRI : CHn_PRI[APL]

Priority vector = ((CHn_GRPRI << 8) + (CHn_PRI[APL] << 5) + CHn_*)

A channel requesting service with the highest priority vector value will receive the next execution slot.

An execution slot is available:

1. immediately if the eDMA is idle
2. when an active channel retires
3. when valid preemption conditions exist

NOTE

Preemption is strictly priority based. Preemption is not bound by a specific group number as defined by CHn_GRPRI.

Channel preemption is enabled on a per-channel basis by setting the CHn_PRI[ECP] field. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher-priority channel. After the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution.

After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended, and the higher-priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted.

A channel's ability to preempt another channel can be disabled by setting CHn_PRI[DPA] to 1. When a channel's preempt ability is disabled, that channel cannot suspend a lower-priority channel's data transfer, regardless of the lower-priority channel's ECP setting. This allows for a pool of low-priority, large-data-moving channels to be defined.

You can configure these low-priority channels to not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true high-priority channel. When you enable round-robin channel arbitration mode (CSR[ERCA] is set to 1), any channel with a priority level equal to 0 (CHn_PRI[APL] = 0) has preemption disabled and cannot preempt another channel.

23.3.5 Clocking

This module has no clocking considerations.

23.3.6 Interrupts

Software can enable the interrupt for each channel for the following events:

1. The major loop is half complete (INTHALF)
2. The major loop is complete (INTMAJOR)
3. A configuration error occurs (EEI)

23.4 External signals

This module has no external signals.

23.5 Initialization

The following sections discuss initialization of the eDMA and programming considerations.

23.5.1 eDMA initialization

To initialize the eDMA:

1. Write to the MP_CSR if a configuration other than the default is wanted.
2. Write the channel priority levels to the CHn_PRI registers and group priority levels to the CHn_GRPRI registers if a configuration other than the default is wanted.
3. Enable error interrupts in the CHn_CSR[EEI] registers if they are wanted.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the CHn_CSR[ERQ] registers.
6. Request channel service via either:
   - Software: setting TCDn_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in Table 296, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, defined by TCDn_SADDR, to the destination, defined by TCD_DADDR, continue until the number of bytes specified by TCDn_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCDn_SADDR, TCDn_DADDR, and TCDn_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, then eDMA executes further post-processing, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 296. TCD control and status (TCDn_CSR) fields

TCDn_CSR field name	Description
START	Control field to start the channel explicitly when using a software-initiated DMA service (automatically cleared by hardware)
EEOP	Control field to enable end-of-packet processing
ESDA	Control field to enable storing of the destination address to system memory after the major loop completes
DREQ	Control field to disable hardware-initiated DMA service requests after major loop completion
BWC	Control field for throttling the bandwidth control of a channel
ESG	Control field to enable the scatter-gather feature
INTHALF	Control field to enable interrupt when major loop is half-complete
INTMAJOR	Control field to enable interrupt when major loop completes

Table 297. Channel control and status (CHn_CSR) fields

CHn_CSR field name	Description
ACTIVE	Status field indicating the channel is currently in execution
DONE	Status field indicating major loop completion (cleared by software when a channel begins execution)
EEI	Control field to enable error interrupts
EARQ	Control field to enable external, asynchronous wakeup event in conjunction with the ERQ field
ERQ	Control field to enable hardware service requests

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

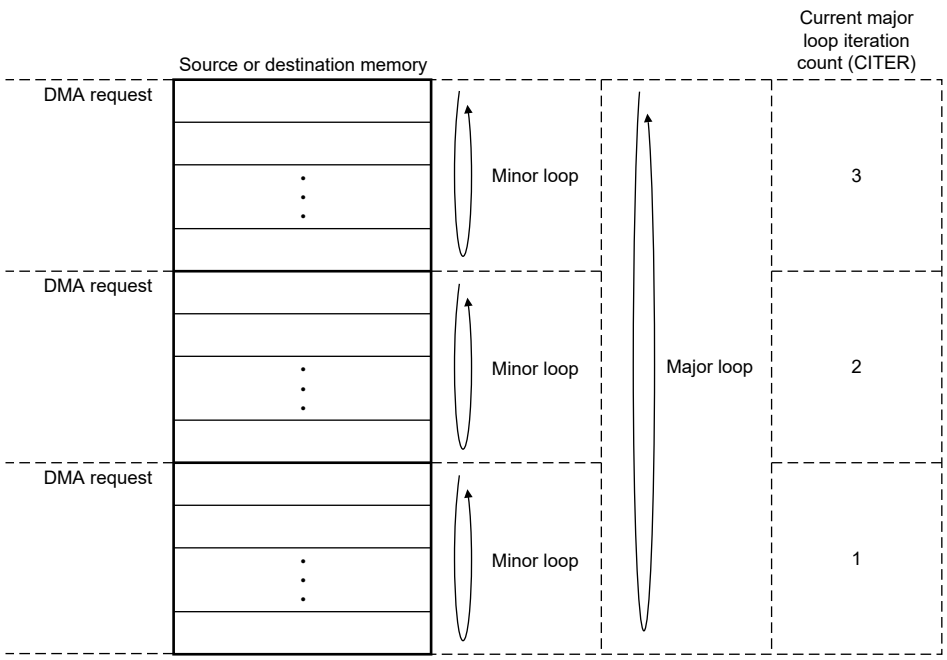


Figure 108. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings are related.

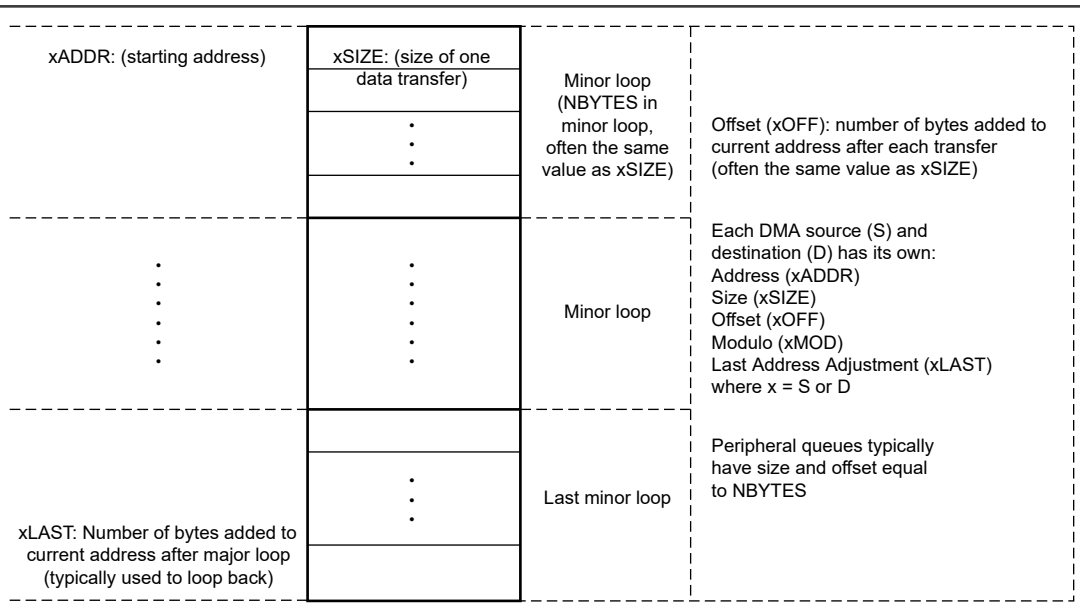


Figure 109. Memory array terms

23.5.2 eDMA arbitration

The eDMA uses a layered arbitration scheme composed of multiple priority levels. The eDMA uses a fixed-priority arbitration scheme with optional round-robin arbitration under specific conditions. The priorities are evaluated in the following order:

Table 298. eDMA arbitration priorities

Priority	Scheme	Description
1 (Highest)	Arbitration group priority	Each channel is assigned an arbitration group via the CHn_GRPRI registers. Priority is given to the highest value (31 being the highest possible value) down to the lowest value (zero, the default).
2	Channel priority	Each channel is assigned a channel priority level via the CHn_PRI registers. The channel priority is a relative priority level within an arbitration group. Priority is given to the highest value (seven being the highest possible value) down to the lowest value (zero, the default). Channel priorities within each arbitration group need not be unique. If multiple channels have the same channel priority level, the channel number will be used to determine priority as defined in row three.
3	Channel number	When two or more channels have the same arbitration group priority and channel priority, the channel number (CHn_NUM) is used to determine the highest priority. Priority is given to the highest channel number. Lowest priority is channel 0. The channel numbers are static and cannot be changed in the programmer's model.
4 (Lowest)	Round-robin	When round-robin is enabled, any channel configured for round-robin operation has lowest priority within an arbitration group. Round-robin is enabled by setting the MP_CSR[ERCA] field to 1. After being enabled, channels with a channel priority of zero (CHn_PRI=0) will use round-robin arbitration. Round-robin arbitration will rotate the channel selection among the channels requesting service with CHn_PRI=0 within the arbitration group. Any non-zero channel within the arbitration group will continue to use fixed-priority arbitration, and if requesting service will be selected over any round-robin channels.

For fixed arbitration, the overall priority can be considered a number composed of three concatenated priority levels: CHn_GRPRI:CHn_PRI:CH_NUM. The largest number has the highest priority and the lowest number has the lowest priority.

For round-robin arbitration, the priority number is CHn_GRPRI:0:X. The module rotates through the CHn_PRI=0 channels requesting service without regard to priority among these channels. Any channel within the arbitration group for which CHn_PRI is greater than 0 will be serviced before the round-robin channels.

23.5.3 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data.

The channel number causing the error is recorded in the Error Status register (CHn_ES). If the error source is not removed before the next activation of the problematic channel, the error is detected and recorded again. Setting the halt after error field, CSR[HAE], will halt the DMA and prevent recurrence of the error.

23.5.4 Arbitration mode considerations

This section discusses arbitration considerations for eDMA.

23.5.4.1 Fixed group arbitration, fixed channel arbitration

In this mode, eDMA selects for execution the channel service request from the highest-priority channel in the highest-priority group. If eDMA is programmed so that the channels within a high-priority group have a high number of requests or large data transfers, that group may consume all the bandwidth of the eDMA controller. That is, no lower-priority groups are serviced if there is always at least one DMA request pending on a channel in the highest-priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly.

23.5.4.2 Fixed group arbitration, round-robin channel arbitration

The highest-priority group with a request is serviced. Lower-priority groups are serviced if no pending requests exist in the higher-priority groups.

Within each group, channels are serviced starting with the highest non-zero channel priority. For all channels with a channel priority programmed to 0, selection begins with the highest channel number requesting service and then rotates through to the lowest channel number requesting service. The round-robin channel arbitration can provide a fairness mechanism to lower-priority channels.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, fixed channel arbitration, but all the channels in the highest-priority group will be serviced. Service latency is short on the highest-priority group, but could potentially be very much longer as the group priority decreases.

23.5.5 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

23.5.5.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the CH*n*_CSR[DONE] field is set to 1 and an interrupt is generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte-wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source, and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
```

```
TCDn_DLAST_SGA= -16
TCDn_CSR[INTMAJ] = 1
TCDn_CSR[START] = 1 (should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] field requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes:
   - CH*n*_CSR[DONE] = 0
   - TCD*n*_CSR[START] = 0
   - CH*n*_CSR[ACTIVE] = 1
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: CH*n*_CSR[ACTIVE] = 0, CH*n*_CSR[DONE] = 1, CH*n*_INT[INT] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

23.5.5.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop, transferring 16 bytes per iteration. After the channel's hardware requests are enabled via the CHn_CSR[ERQ] register field, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware (eDMA peripheral) requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: CH*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, CH*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:

a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
h. Write 32 bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: CH*n*_CSR[ACTIVE] = 0.
8. The channel retires, which concludes one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware (eDMA peripheral) requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: CH*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, CH*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: Channel TCD data from local memory to internal register file.
13. The source-to-destination transfers are executed as follows:
    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
    b. Write 32 bits to location 0x2010 → first iteration of the minor loop.
    c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
    d. Write 32 bits to location 0x2014 → second iteration of the minor loop.
    e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
    f. Write 32 bits to location 0x2018 → third iteration of the minor loop.
    g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
    h. Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: CH*n*_CSR[ACTIVE] = 0, CH*n*_CSR[DONE] = 1, CH*n*_INT[INT] = 1.
16. The channel retires, which concludes with the major loop complete. The eDMA goes idle or services the next channel.

23.5.5.3 Using the modulo feature

The modulo feature of the eDMA allows implementation of a circular data queue in which the size of the queue is a power of 2. xMOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature. Modulo addressing applies to cases where the minor loop offset is enabled; that is, the upper address bits remain the same after the minor loop offset is added to the source or destination address.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value but the 28 upper address bits (0x1234567*x*) retain their original value. In this example, the source address is set to 0x12345670, the offset is set to four bytes, and the MOD field is set to four, which allows for a 2^4 byte (16 byte) queue size.

Table 299. Modulo example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

23.5.6 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

23.5.6.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software-initiated service requests.

1. The first method is to read the TCDn_CITER field and test for a change.
2. The second method, extracted from the sequence shown below, is to test the TCDn_CSR[START] field and the CHn_CSR[ACTIVE] field. The minor-loop-complete condition is indicated by both fields reading 0 after TCDn_CSR[START] is set to 1. Polling the CHn_CSR[ACTIVE] field only may be inconclusive because the active status may be missed if the channel execution is short in duration.

The CHn_CSR and TCDn_CSR status fields execute the following sequence for a software-activated channel:

Stage	TCDn_CSR field	CHn_CSR fields		State
	START	ACTIVE	DONE	
1	1	0	0	Initiate channel service request via software.
2	0	1	0	Channel is executing.
3a	0	0	0	Channel has completed the minor loop and is idle.
3b	0	0	1	Channel has completed the major loop and is idle.

The best method to test for minor-loop completion when using hardware-initiated (that is, peripheral-initiated) service requests is to read the TCDn_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status fields execute the following sequence for a hardware-activated channel:

Stage	TCDn_CSR field	CHn_CSR fields		State
	START	ACTIVE	DONE	
1	0	0	0	Initiate channel service request via hardware (peripheral request asserted).
2	0	1	0	Channel is executing.
3a	0	0	0	Channel has completed the minor loop and is idle.
3b	0	0	1	Channel has completed the major loop and is idle.

For both activation types, the major-loop-complete status is explicitly indicated via the CHn_CSR[DONE] field.

The TCDn_CSR[START] field is cleared to 0 automatically when the channel begins execution, regardless of how the channel activates.

23.5.6.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCDn_SADDR, TCDn_DADDR, and TCDn_NBYTES values if they are read when a channel executes. The true values of SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file, and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES (which decrements to zero as the transfer progresses), can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

23.5.6.3 Checking channel preemption status

A preemptive situation is one in which a preempt-enabled channel is executing and a higher-priority request becomes active. When round-robin channel arbitration mode is enabled, all channels with their channel priority set to 0 lose their preempt ability. Channel priorities of 0 are treated as equal, that is, they are constantly rotating, when round-robin arbitration mode is enabled.

The CHn_CSR[ACTIVE] field for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended when the preempting channel executes one major loop iteration. If two CHn_CSR[ACTIVE] fields are set simultaneously in the global TCD map, a higher-priority channel is actively preempting a lower-priority channel.

23.5.7 Channel linking

Channel linking (or chaining) is a mechanism in which one channel sets the TCDn_CSR[START] field of another channel (or itself), thus initiating a service request for that channel. When properly enabled, the eDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCDn_CITER[ELINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, using an initial field setting of:

```
MP_CSR[GCLC] = 1
TCDn_CITER[ELINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJORELINK] = 1
TCDn_CSR[MAJORLINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] field
2. Minor loop done → set TCD12_CSR[START] field
3. Minor loop done → set TCD12_CSR[START] field
4. Minor loop done, major loop done→ set TCD7_CSR[START] field

When minor loop linking is enabled (TCDn_CITER[ELINK] = 1), the TCDn_CITER[CITER] field uses a nine-bit vector to form the current iteration count. When minor loop linking is disabled (TCDn_CITER[ELINK] = 0), the TCDn_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCDn_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

NOTE

The TCDn_CITER[ELINK] field and the TCDn_BITER[ELINK] field must be equal — if they are not, a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop halfway done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, that is, use another channel's TCD, at the end of a loop.

Table 300. Channel linking parameters

Wanted link behavior	TCD control field name	Description
Link at end of minor loop	TCDn_CITER[ELINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	TCDn_CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of major loop	TCDn_CSR[MAJORELINK]	Enable channel-to-channel linking on major loop completion
	TCDn_CSR[MAJORLINKCH]	Link channel number when linking at end of major loop

23.5.8 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

23.5.8.1 Dynamically changing the channel priority

To change group or channel priority levels:

1. Halt the DMA by writing 1 to the CSR[HALT] field.
2. Change the group or channel priorities as wanted.
3. Enable normal DMA operations by writing 0 to the CSR[HALT] field.

23.5.8.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCDn_CSR[MAJORELINK] field during channel execution (see the diagram in TCD structure). This field is read from the TCD local memory at the end of channel execution, thus allowing you to enable the feature during channel execution.

Because you are allowed to change the configuration during execution, you need a coherency model. Consider the scenario where you attempt to execute a dynamic channel link by enabling the TCDn_CSR[MAJORELINK] field at the same time the eDMA engine is retiring the channel. TCDn_CSR[MAJORELINK] would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

We recommend that you use the following coherency model when executing a dynamic channel link request.

1. Write 1 to the TCDn_CSR[MAJORELINK] field.
2. Read back the TCDn_CSR[MAJORELINK] field.
3. Test the TCDn_CSR[MAJORELINK] request status:
   - If TCDn_CSR[MAJORELINK] = 1, the dynamic link attempt was successful.
   - If TCDn_CSR[MAJORELINK] = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCDn_CSR[MAJORELINK] field to 0 on any writes to a channel's TCDn_CSR[7:0] after that channel's CHn_CSR[DONE] field is set to 1, indicating the major loop is complete.

NOTE

You must clear the CHn_CSR[DONE] field to 0 before writing to the TCDn_CSR[MAJORELINK] field. The CHn_CSR[DONE] field is cleared to 0 automatically by the eDMA engine after a channel begins execution.

23.5.8.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in the eDMA programmer's model, thus replacing the current descriptor.

Because you are allowed to change the configuration during execution, you need a coherency model. Consider the scenario where you attempt to execute a dynamic scatter/gather operation by enabling the TCDn_CSR[ESG] field at the same time the eDMA engine is retiring the channel. The TCDn_CSR[ESG] field would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods are recommended for executing a dynamic scatter/gather request. Whenever the TCDn_CSR is written, the TCD local memory controller forces the TCDn_CSR[ESG] field to 0 on any writes to a channel's TCDn_CSR[7:0] after that channel's CHn_CSR[DONE] field has been set to 1, indicating the major loop is complete. If attempting to set the ESG, ensure the DONE field is cleared to 0.

NOTE

You must clear the CHn_CSR[DONE] field to 0 before writing the TCDn_CSR[MAJORELINK] or TCDn_CSR[ESG] fields. The CHn_CSR[DONE] field is cleared to 0 automatically by the eDMA engine after a channel begins execution and is set to 1 upon major loop completion.

23.5.8.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCDn_CSR[MAJORELINK] field is 0, the TCDn_CSR[MAJORLINKCH] field is not used by the eDMA. In this case, the TCDn_CSR[MAJORLINKCH] bits may be used for other purposes. This method uses the TCDn_CSR[MAJORLINKCH] field as a TCDn_CSR identification (ID).

When the descriptors are built, write a unique TCDn_CSR ID in the TCDn_CSR[MAJORLINKCH] field for each TCDn_CSR associated with a channel using dynamic scatter/gather.

1. Write a 1 to the TCDn_CSR[DREQ] field. Should a dynamic scatter/gather attempt fail, setting the TCDn_CSR[DREQ] field to 1 will prevent future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST final offset value.
2. Write the TCDn_DLAST_SGA field with the scatter/gather address.
3. Write a 1 to the TCDn_CSR[ESG] field.
4. Read back the 16-bit TCDn_CSR control/status field.
5. Test the TCDn_CSR[ESG] request status and TCDn_CSR[MAJORLINKCH] value:

- If ESG = 1, the dynamic scatter/gather attempt was successful.
- If ESG = 0 and the MAJORLINKCH (ID) did not change, the dynamic scatter/gather attempt was not successful (the channel was already retiring).
- If ESG = 0 and the MAJORLINKCH (ID) changed, the dynamic scatter/gather attempt was successful (the new TCDn_CSR's ESG value cleared the ESG field to 0).

23.5.8.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCDn_DLAST_SGA field as a TCD identification (ID).

1. Write a 1 to the TCDn_CSR[DREQ] field. Should a dynamic scatter/gather attempt fail, setting the DREQ field to 1 will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST final offset value.
2. Write the TCDn_DLAST_SGA field with the scatter/gather address.
3. Write a 1 to the TCDn_CSR[ESG] field.
4. Read back the TCDn_CSR[ESG] field.
5. Test the TCDn_CSR[ESG] request status:
   - If ESG = 1, the dynamic scatter/gather attempt was successful.
   - If ESG = 0, read the 32-bit TCDn_DLAST_SGA field.
   - If ESG = 0 and the TCDn_DLAST_SGA did not change, the dynamic scatter/gather attempt was not successful (the channel was already retiring).
   - If ESG = 0 and the TCDn_DLAST_SGA changed, the dynamic scatter/gather attempt was successful (the new TCDn_CSR's ESG value cleared the ESG field to 0).

23.5.9 Suspend/resume a DMA channel with active hardware service requests

The DMA allows you to move data from memory or peripheral registers to another location in memory or to peripheral registers without CPU interaction. After the DMA and peripherals are configured and active, it is rare but supported to suspend a peripheral's service request dynamically. In this scenario, there are certain restrictions to disabling a DMA hardware service request. For coherency, you must follow a specific procedure. This section provides guidance on how to coherently suspend and resume a Direct Memory Access (DMA) channel when the DMA is triggered by a slave module such as the Serial Peripheral Interface (DSPI), Sigma Delta Analog to Digital Convertor (SDADC), or other module.

23.5.9.1 Suspend an active DMA channel

To suspend an active DMA channel:

1. Stop the DMA service request at the peripheral first. Confirm it has been disabled by reading back the appropriate register in the peripheral.
2. Check the DMA's Hardware Request Status (MP_HRS) to ensure there is no service request to the DMA channel being suspended. Then disable the hardware service request by clearing the ERQ field to 0 on the appropriate DMA channel.

For example, assume the DSPI is set as a master for transmitting data via a DMA service request when the TXFIFO has an empty slot. The DMA will transfer the next command and data to the TXFIFO upon the request. If you need to suspend the DMA/DSPI transfer loop, perform the following steps:

1. Disable the DMA service request at the source by writing 0 to DSPI_RSER[TFFF_RE]. Confirm that DSPI_RSER[TFFF_RE] is 0.
2. Ensure there is no DMA service request from the DSPI by verifying that MP_HRS[HRS] is 0 for the appropriate channel. If no service request is present, disable the DMA channel by clearing the channel's ERQ field to 0. If a service request is present, wait until the request has been processed and the HRS field reads 0.

23.5.9.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting its ERQ field to 1.
2. Enable the DMA service request at the peripheral.

23.6 Memory map/register definition

The eDMA programming model is partitioned into three parts:

1. The first part defines a number of registers providing overall control functions and is known as the management page.
2. The second part corresponds to the channel (CH) control, status, and configuration.
3. The third part corresponds to the local TCD memory.

TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the data movement operation. Each TCDn definition is presented as 11 registers of 16 or 32 bits. See DMA TCD register descriptions for details.

TCD initialization

Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

TCD structure

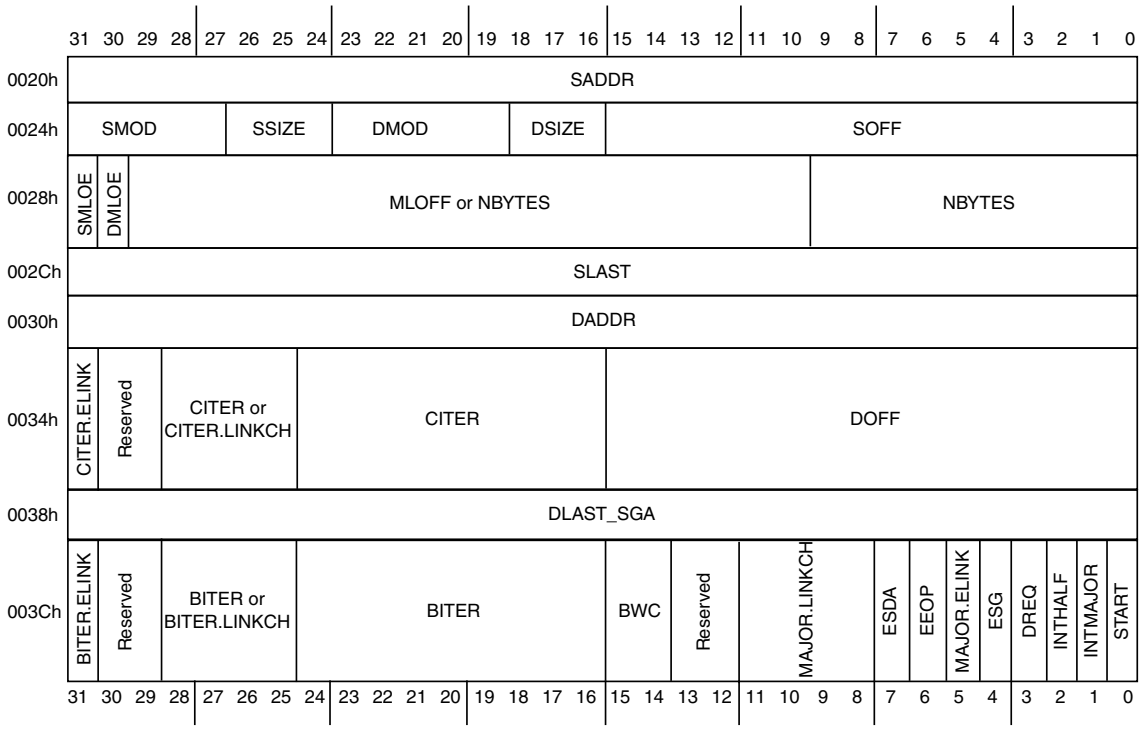


Figure 110. TCD structure

Accesses to reserved memory and fields

- Reading reserved fields in a register returns the value of zero.
- Writes to reserved fields in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

23.6.1 DMA MP register descriptions

23.6.1.1 MP memory map

eDMA_0_MP base address: 4008_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Management Page Control (MP_CSR)	32	RW	0031_0000h
4h	Management Page Error Status (MP_ES)	32	R	0000_0000h
8h	Management Page Interrupt Request Status (MP_INT)	32	R	0000_0000h
Ch	Management Page Hardware Request Status (MP_HRS)	32	R	0000_0000h
100h - 13Ch	Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI)	32	RW	0000_0000h

23.6.1.2 Management Page Control (MP_CSR)

Offset

Register	Offset
MP_CSR	0h

Function

The Management Page Control register defines the basic operating configuration of the DMA.

Arbitration uses a two-tier priority system; group and channel priority. The eDMA assigns each channel to a priority group. Group arbitration is fixed-priority and cannot be changed. Channel arbitration uses fixed priority and may be configured to use a selective round-robin scheme for specified channels within each priority group. For fixed-priority arbitration, eDMA selects for execution the highest priority channel requesting service in the highest priority arbitration group.

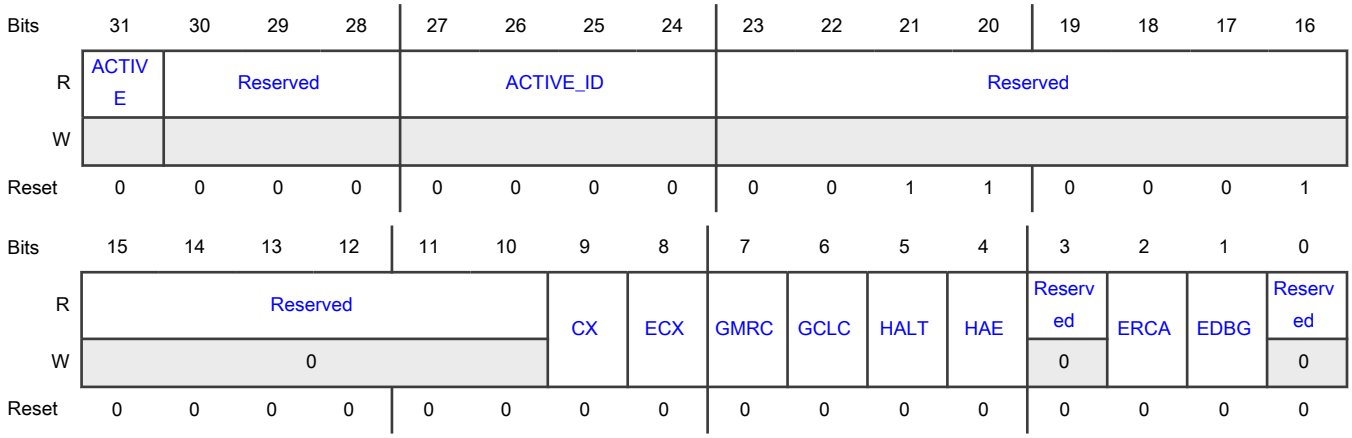
The channel priority registers assign the relative priorities within each arbitration group; see CHn_PRI. All channels with a non-zero CHn_PRI value use fixed-priority arbitration.

When you enable round-robin arbitration, all channels with channel priority set to zero do not have a priority and, of those channels requesting service, are cycled through (from high to low channel number) without regard to priority relative to each other within the same priority group. Any channel with a non-zero CHn_PRI value automatically has a higher priority over the round-robin channels. A channel's priority group is assigned in Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI).

NOTE

For correct operation, changes to the MP_CSR[ERCA, GCLC, GMRC] fields must be performed when the DMA channels are inactive; that is, when the MP_CSR[ACTIVE] field is 0.

Diagram



Fields

Field	Function
31 ACTIVE	DMA Active Status 0b - eDMA is idle 1b - eDMA is executing a channel
30-28 —	Reserved
27-24 ACTIVE_ID	Active Channel ID This field identifies the channel number that is executing when the ACTIVE bit is 1.
23-16 —	Reserved
15-10 —	Reserved
9 CX	Cancel Transfer When set to 1, this field cancels the remaining data transfer, stops the executing channel, and forces the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. CX clears itself to 0 after the cancel has been honored. This cancel retires the channel normally as if the minor loop had been completed. 0b - Normal operation 1b - Cancel the remaining data transfer
8 ECX	Cancel Transfer With Error Cancellation of the remaining data transfer is similar to that of the CX field. Execution of the channel is stopped and the minor loop is forced to finish. The cancellation takes effect after the last write of the current read/write sequence. The ECX field clears itself to 0 after the cancel is honored. In addition to

Table continues on the next page...

Table continued from the previous page...

Field	Function
	cancelling the transfer, ECX treats the cancel as an error condition, thus updating Management Page Error Status (MP_ES) and generating an optional error interrupt. 0b - Normal operation 1b - Cancel the remaining data transfer
7 GMRC	Global Master ID Replication Control **NOTE** If master ID replication is disabled, the nonsecure, user protection level for DMA transfers is used. 0b - Master ID replication disabled for all channels 1b - Master ID replication available and controlled by each channel's CHn_SBR[EMI] setting
6 GCLC	Global Channel Linking Control 0b - Channel linking disabled for all channels 1b - Channel linking available and controlled by each channel's link settings
5 HALT	Halt DMA Operations This field stalls the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this field is cleared to 0. 0b - Normal operation 1b - Stall the start of any new channels
4 HAE	Halt After Error When this field is set to 1, any error causes the HALT field to be set to 1. Then all service requests are ignored until the HALT field is cleared to 0. 0b - Normal operation 1b - Any error causes the HALT field to be set to 1
3 —	Reserved
2 ERCA	Enable Round Robin Channel Arbitration 0b - Round-robin channel arbitration disabled. Fixed priority arbitration used for channel selection 1b - Round-robin channel arbitration enabled. Round-robin arbitration used for channel selection
1 EDBG	Enable Debug When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. DMA resumes channel execution when the system exits debug mode or clears the EDBG field to 0. 0b - Debug mode disabled. When in debug mode, the DMA continues to operate

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Debug mode is enabled. When in debug mode, the DMA stalls the start of a new channel
0 —	Reserved

23.6.1.3 Management Page Error Status (MP_ES)

Offset

Register	Offset
MP_ES	4h

Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- An illegal setting in the transfer control descriptor
- An error termination to a bus master read or write cycle
- An uncorrectable error that occurred when the device was accessing the TCD SRAM
- A "cancel transfer with error" request was made via the corresponding cancel transfer field or input signal

Upon any error condition, the software must initialize the TCD of the channel that contains the error, as it is in an incomplete state after an error. See Fault reporting and handling for more details.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	Reserved			ERRCHN				0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							ECX	SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Valid Logical OR of all CHn_ES[ERR] status fields.

Table continues on the next page...

Table continued from the previous page...

Field	Function
VLD	0b - No CHn_ES[ERR] fields are set to 1 1b - At least one CHn_ES[ERR] field is set to 1, indicating a valid error exists that software has not cleared
30-28 —	Reserved
27-24 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error or last recorded error-canceled transfer.
23-9 —	Reserved
8 ECX	Transfer Canceled The ECX operation is a management page function. When employed, the targeted channel's CHn_ES register reports an unspecified error; that is, only the CHn_ES[ERR] field is set to 1. The management page has full view of the error condition. 0b - No canceled transfers 1b - Last recorded entry was a canceled transfer by the error cancel transfer input
7 SAE	Source Address Error When this field is 1, it indicates that TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SADDR field
6 SOE	Source Offset Error When this field is 1, it indicates that TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SOFF field
5 DAE	Destination Address Error When this field is 1, it indicates that TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DADDR field
4 DOE	Destination Offset Error When this field is 1, it indicates that TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DOFF field
3	NBYTES/CITER Configuration Error

Table continues on the next page...

Table continued from the previous page...

Field	Function
NCE	This error indicates that one of the following has occurred: • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE] • TCDn_CITER[CITER] is equal to zero • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK] 0b - No NBYTES/CITER configuration error 1b - The last recorded error was NBYTES equal to zero or a CITER not equal to BITER error. Last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields
2 SGE	Scatter/Gather Configuration Error When this field is 1, it indicates that TCDn_DLAST_SGA is not on a 32-byte boundary. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. 0b - No scatter/gather configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DLAST_SGA field
1 SBE	Source Bus Error 0b - No source bus error 1b - Last recorded error was a bus error on a source read
0 DBE	Destination Bus Error 0b - No destination bus error 1b - Last recorded error was a bus error on a destination write

23.6.1.4 Management Page Interrupt Request Status (MP_INT)

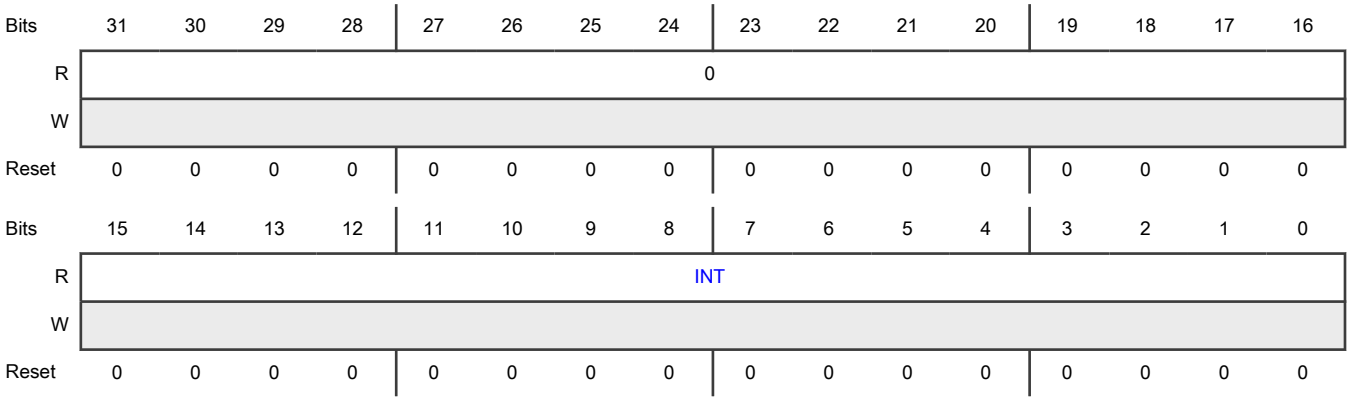
Offset

Register	Offset
MP_INT	8h

Function

This register shows the current state of the interrupt service requests for all eDMA channels.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 INT	Interrupt Request Status The INT register presents the interrupt request status for each eDMA channel. Depending on the appropriate field setting in the transfer control descriptors, the eDMA engine generates an interrupt on data transfer completion or an error condition. The eDMA routes channel interrupt requests to the interrupt controller. During the interrupt service routine associated with any given channel, it is the software's responsibility to clear the appropriate field in the channel’s interrupt request register, CHn_INT, thus negating the interrupt request. 0b - Interrupt request for corresponding channel not present 1b - Interrupt request for corresponding channel present

23.6.1.5 Management Page Hardware Request Status (MP_HRS)

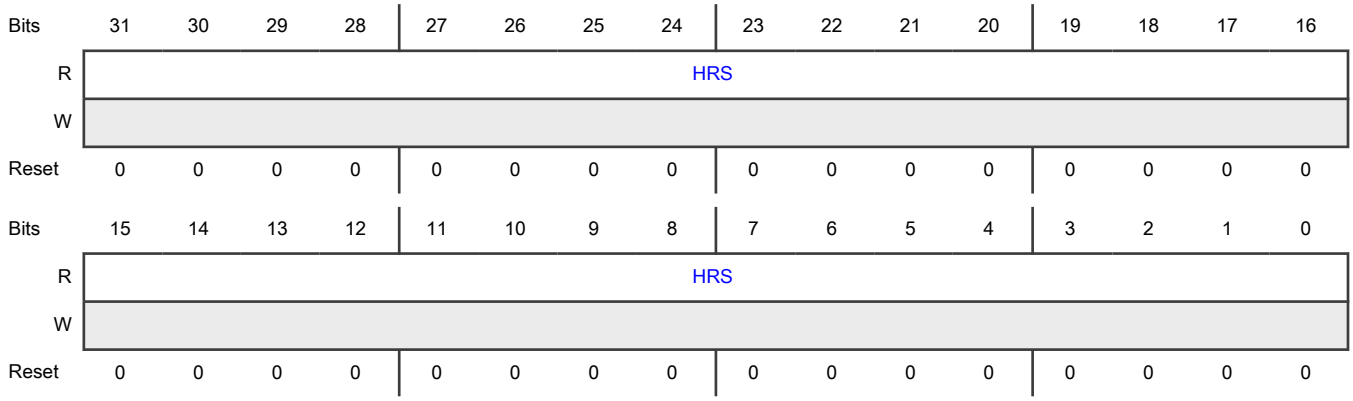
Offset

Register	Offset
MP_HRS	Ch

Function

The hardware request status register (HRS) shows the current state of the hardware service request signaling as seen by eDMA's arbitration logic.

Diagram



Fields

Field	Function
31-0 HRS	Hardware Request Status The HRS bit for its respective channel remains asserted for the period when a hardware request is present on the channel. 0b - Hardware service request for corresponding channel is not present 1b - Hardware service request for corresponding channel is present

23.6.1.6 Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI)

Offset

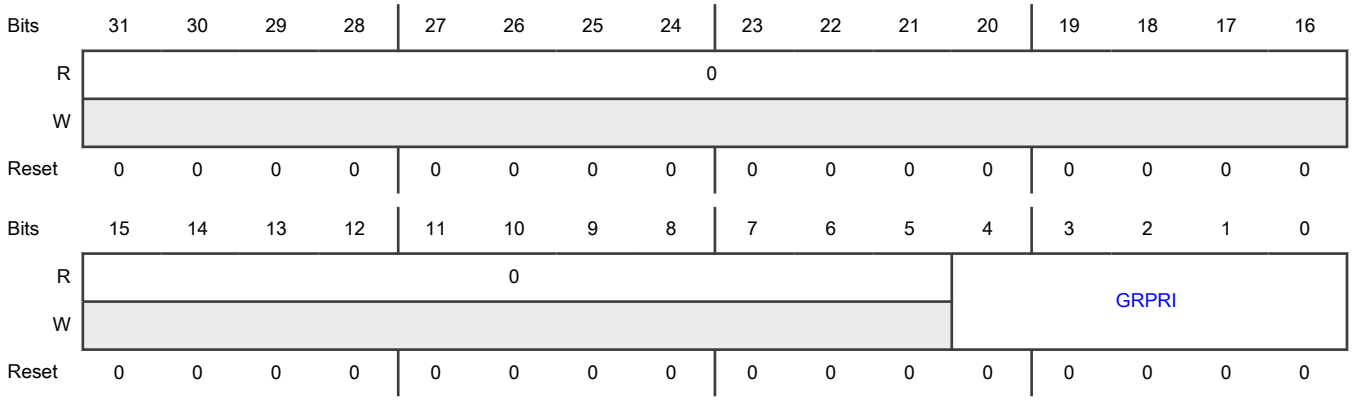
For n = 0 to 15:

Register	Offset
CHn_GRPRI	100h + (n × 4h)

Function

The contents of this register define the arbitration group associated with each channel. Using a fixed-priority group arbitration scheme, eDMA evaluates the arbitration group priorities by numeric value from highest group number to lowest; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. The range of the group priority values is limited to the values of 0 through 31. Within each arbitration group, the channel priority assignment CH*n*_PRI determines the highest-priority channel.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 GRPRI	Arbitration Group For Channel n Fixed-priority arbitration group number.

23.6.2 DMA MP register descriptions

23.6.2.1 MP memory map

eDMA_1_MP base address: 400A_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Management Page Control (MP_CSR)	32	RW	0031_0000h
4h	Management Page Error Status (MP_ES)	32	R	0000_0000h
8h	Management Page Interrupt Request Status (MP_INT)	32	R	0000_0000h
Ch	Management Page Hardware Request Status (MP_HRS)	32	R	0000_0000h
100h - 13Ch	Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI)	32	RW	0000_0000h

23.6.2.2 Management Page Control (MP_CSR)

Offset

Register	Offset
MP_CSR	0h

Function

The Management Page Control register defines the basic operating configuration of the DMA.

Arbitration uses a two-tier priority system; group and channel priority. The eDMA assigns each channel to a priority group. Group arbitration is fixed-priority and cannot be changed. Channel arbitration uses fixed priority and may be configured to use a selective round-robin scheme for specified channels within each priority group. For fixed-priority arbitration, eDMA selects for execution the highest priority channel requesting service in the highest priority arbitration group.

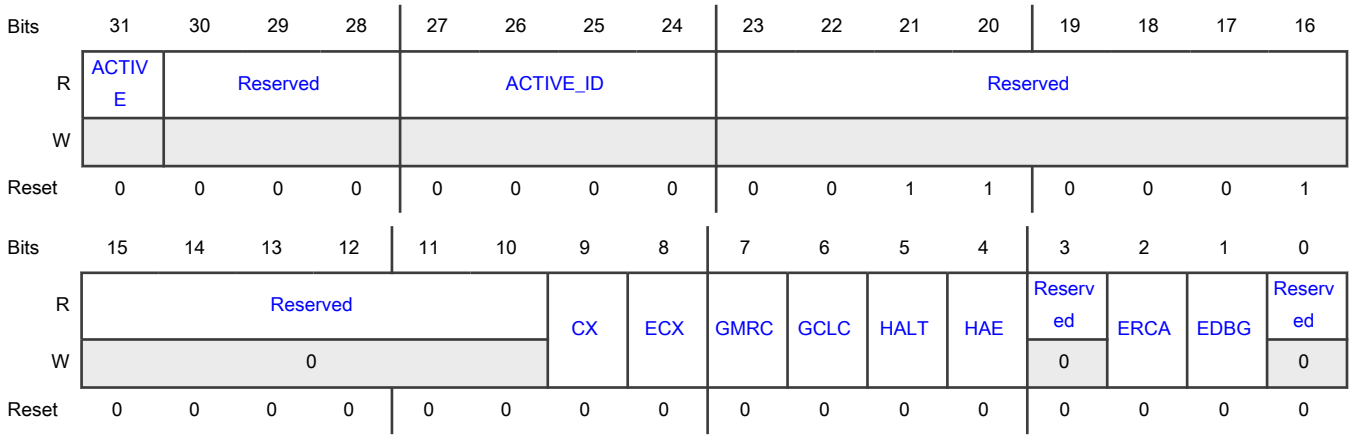
The channel priority registers assign the relative priorities within each arbitration group; see CHn_PRI. All channels with a non-zero CHn_PRI value use fixed-priority arbitration.

When you enable round-robin arbitration, all channels with channel priority set to zero do not have a priority and, of those channels requesting service, are cycled through (from high to low channel number) without regard to priority relative to each other within the same priority group. Any channel with a non-zero CHn_PRI value automatically has a higher priority over the round-robin channels. A channel's priority group is assigned in Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI).

NOTE

For correct operation, changes to the MP_CSR[ERCA, GCLC, GMRC] fields must be performed when the DMA channels are inactive; that is, when the MP_CSR[ACTIVE] field is 0.

Diagram



Fields

Field	Function
31 ACTIVE	DMA Active Status 0b - eDMA is idle 1b - eDMA is executing a channel
30-28 —	Reserved
27-24 ACTIVE_ID	Active Channel ID This field identifies the channel number that is executing when the ACTIVE bit is 1.
23-16	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
15-10 —	Reserved
9 CX	Cancel Transfer When set to 1, this field cancels the remaining data transfer, stops the executing channel, and forces the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. CX clears itself to 0 after the cancel has been honored. This cancel retires the channel normally as if the minor loop had been completed. 0b - Normal operation 1b - Cancel the remaining data transfer
8 ECX	Cancel Transfer With Error Cancellation of the remaining data transfer is similar to that of the CX field. Execution of the channel is stopped and the minor loop is forced to finish. The cancellation takes effect after the last write of the current read/write sequence. The ECX field clears itself to 0 after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating Management Page Error Status (MP_ES) and generating an optional error interrupt. 0b - Normal operation 1b - Cancel the remaining data transfer
7 GMRC	Global Master ID Replication Control **NOTE** If master ID replication is disabled, the nonsecure, user protection level for DMA transfers is used. 0b - Master ID replication disabled for all channels 1b - Master ID replication available and controlled by each channel's CHn_SBR[EMI] setting
6 GCLC	Global Channel Linking Control 0b - Channel linking disabled for all channels 1b - Channel linking available and controlled by each channel's link settings
5 HALT	Halt DMA Operations This field stalls the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this field is cleared to 0. 0b - Normal operation 1b - Stall the start of any new channels
4 HAE	Halt After Error

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When this field is set to 1, any error causes the HALT field to be set to 1. Then all service requests are ignored until the HALT field is cleared to 0. 0b - Normal operation 1b - Any error causes the HALT field to be set to 1
3 —	Reserved
2 ERCA	Enable Round Robin Channel Arbitration 0b - Round-robin channel arbitration disabled. Fixed priority arbitration used for channel selection 1b - Round-robin channel arbitration enabled. Round-robin arbitration used for channel selection
1 EDBG	Enable Debug When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. DMA resumes channel execution when the system exits debug mode or clears the EDBG field to 0. 0b - Debug mode disabled. When in debug mode, the DMA continues to operate 1b - Debug mode is enabled. When in debug mode, the DMA stalls the start of a new channel
0 —	Reserved

23.6.2.3 Management Page Error Status (MP_ES)

Offset

Register	Offset
MP_ES	4h

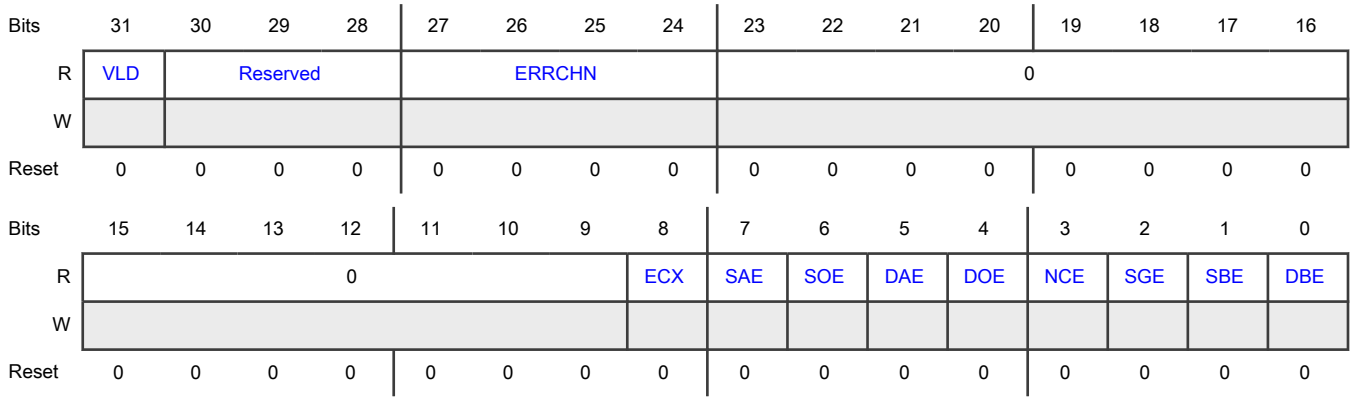
Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- An illegal setting in the transfer control descriptor
- An error termination to a bus master read or write cycle
- An uncorrectable error that occurred when the device was accessing the TCD SRAM
- A "cancel transfer with error" request was made via the corresponding cancel transfer field or input signal

Upon any error condition, the software must initialize the TCD of the channel that contains the error, as it is in an incomplete state after an error. See Fault reporting and handling for more details.

Diagram



Fields

Field	Function
31 VLD	Valid Logical OR of all CHn_ES[ERR] status fields. 0b - No CHn_ES[ERR] fields are set to 1 1b - At least one CHn_ES[ERR] field is set to 1, indicating a valid error exists that software has not cleared
30-28 —	Reserved
27-24 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error or last recorded error-canceled transfer.
23-9 —	Reserved
8 ECX	Transfer Canceled The ECX operation is a management page function. When employed, the targeted channel's CHn_ES register reports an unspecified error; that is, only the CHn_ES[ERR] field is set to 1. The management page has full view of the error condition. 0b - No canceled transfers 1b - Last recorded entry was a canceled transfer by the error cancel transfer input
7 SAE	Source Address Error When this field is 1, it indicates that TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SADDR field
6	Source Offset Error

Table continues on the next page...

Table continued from the previous page...

Field	Function
SOE	When this field is 1, it indicates that TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SOFF field
5 DAE	Destination Address Error When this field is 1, it indicates that TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DADDR field
4 DOE	Destination Offset Error When this field is 1, it indicates that TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DOFF field
3 NCE	NBYTES/CITER Configuration Error This error indicates that one of the following has occurred: • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE] • TCDn_CITER[CITER] is equal to zero • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK] 0b - No NBYTES/CITER configuration error 1b - The last recorded error was NBYTES equal to zero or a CITER not equal to BITER error. Last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields
2 SGE	Scatter/Gather Configuration Error When this field is 1, it indicates that TCDn_DLAST_SGA is not on a 32-byte boundary. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. 0b - No scatter/gather configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DLAST_SGA field
1 SBE	Source Bus Error 0b - No source bus error 1b - Last recorded error was a bus error on a source read
0 DBE	Destination Bus Error 0b - No destination bus error 1b - Last recorded error was a bus error on a destination write

23.6.2.4 Management Page Interrupt Request Status (MP_INT)

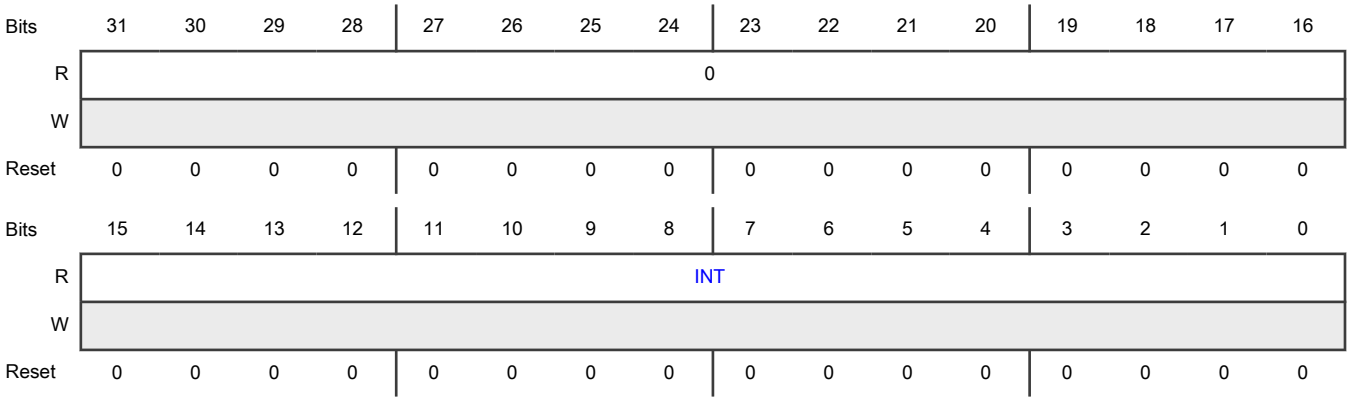
Offset

Register	Offset
MP_INT	8h

Function

This register shows the current state of the interrupt service requests for all eDMA channels.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 INT	Interrupt Request Status The INT register presents the interrupt request status for each eDMA channel. Depending on the appropriate field setting in the transfer control descriptors, the eDMA engine generates an interrupt on data transfer completion or an error condition. The eDMA routes channel interrupt requests to the interrupt controller. During the interrupt service routine associated with any given channel, it is the software's responsibility to clear the appropriate field in the channel's interrupt request register, CHn_INT, thus negating the interrupt request. 0b - Interrupt request for corresponding channel not present 1b - Interrupt request for corresponding channel present

23.6.2.5 Management Page Hardware Request Status (MP_HRS)

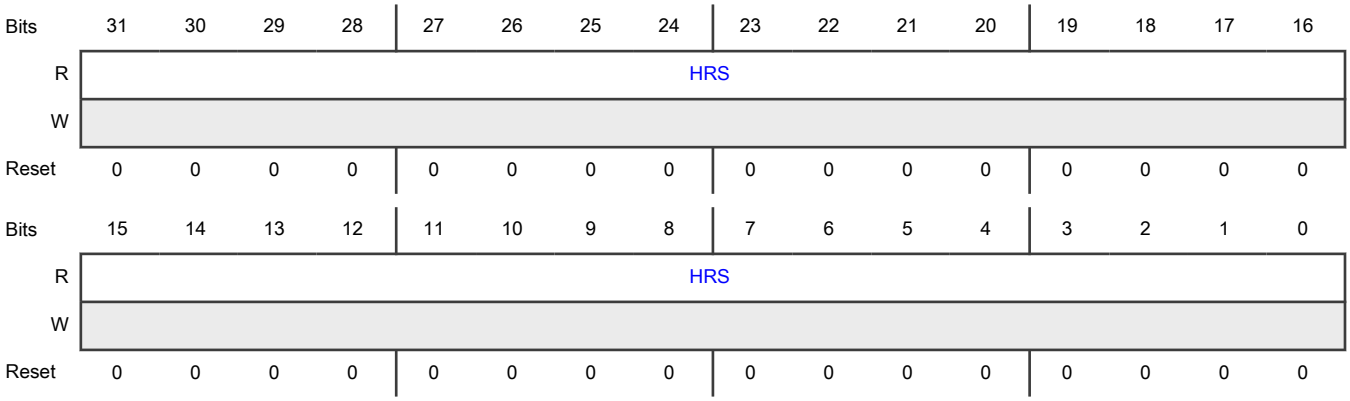
Offset

Register	Offset
MP_HRS	Ch

Function

The hardware request status register (HRS) shows the current state of the hardware service request signaling as seen by eDMA's arbitration logic.

Diagram



Fields

Field	Function
31-0 HRS	Hardware Request Status The HRS bit for its respective channel remains asserted for the period when a hardware request is present on the channel. 0b - Hardware service request for corresponding channel is not present 1b - Hardware service request for corresponding channel is present

23.6.2.6 Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI)

Offset

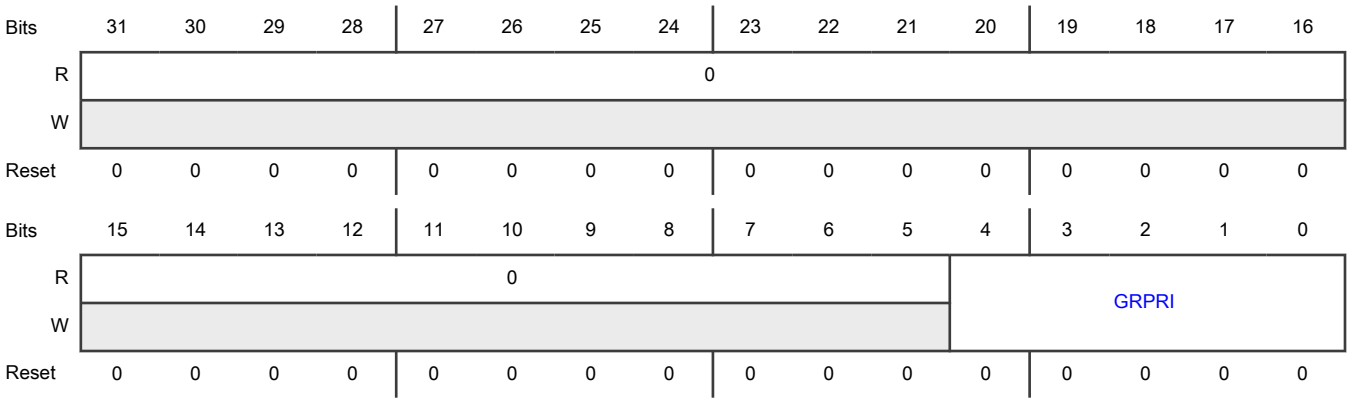
For n = 0 to 15:

Register	Offset
CHn_GRPRI	100h + (n × 4h)

Function

The contents of this register define the arbitration group associated with each channel. Using a fixed-priority group arbitration scheme, eDMA evaluates the arbitration group priorities by numeric value from highest group number to lowest; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. The range of the group priority values is limited to the values of 0 through 31. Within each arbitration group, the channel priority assignment CH*n*_PRI determines the highest-priority channel.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 GRPRI	Arbitration Group For Channel n Fixed-priority arbitration group number.

23.6.3 DMA TCD register descriptions

23.6.3.1 TCD memory map

eDMA_0_TCD base address: 4008_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h - F000h	Channel Control and Status (CH0_CSR - CH15_CSR)	32	RW	0000_0000h
4h - F004h	Channel Error Status (CH0_ES - CH15_ES)	32	RW	0000_0000h
8h - F008h	Channel Interrupt Status (CH0_INT - CH15_INT)	32	RW	0000_0000h
Ch - F00Ch	Channel System Bus (CH0_SBR - CH15_SBR)	32	RW	0000_0006h
10h - F010h	Channel Priority (CH0_PRI - CH15_PRI)	32	RW	0000_0000h
14h - F014h	Channel Multiplexor Configuration (CH0_MUX - CH15_MUX)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
20h - F020h	TCD Source Address (TCD0_SADDR - TCD15_SADDR)	32	RW	See section
24h - F024h	TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)	16	RW	See section
26h - F026h	TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)	16	RW	See section
28h - F028h	TCD Transfer Size Without Minor Loop Offsets (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)	32	RW	See section
28h - F028h	TCD Transfer Size with Minor Loop Offsets (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)	32	RW	See section
2Ch - F02Ch	TCD Last Source Address Adjustment / Store DADDR Address (TCD0_SLAST_SDA - TCD15_SLAST_SDA)	32	RW	See section
30h - F030h	TCD Destination Address (TCD0_DADDR - TCD15_DADDR)	32	RW	See section
34h - F034h	TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)	16	RW	See section
36h - F036h	TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)	16	RW	See section
36h - F036h	TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)	16	RW	See section
38h - F038h	TCD Last Destination Address Adjustment / Scatter Gather Address (TCD0_DLAST_SGA - TCD15_DLAST_SGA)	32	RW	See section
3Ch - F03Ch	TCD Control and Status (TCD0_CSR - TCD15_CSR)	16	RW	See section
3Eh - F03Eh	TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)	16	RW	See section
3Eh - F03Eh	TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)	16	RW	See section

23.6.3.2 Channel Control and Status (CH0_CSR - CH15_CSR)

Offset

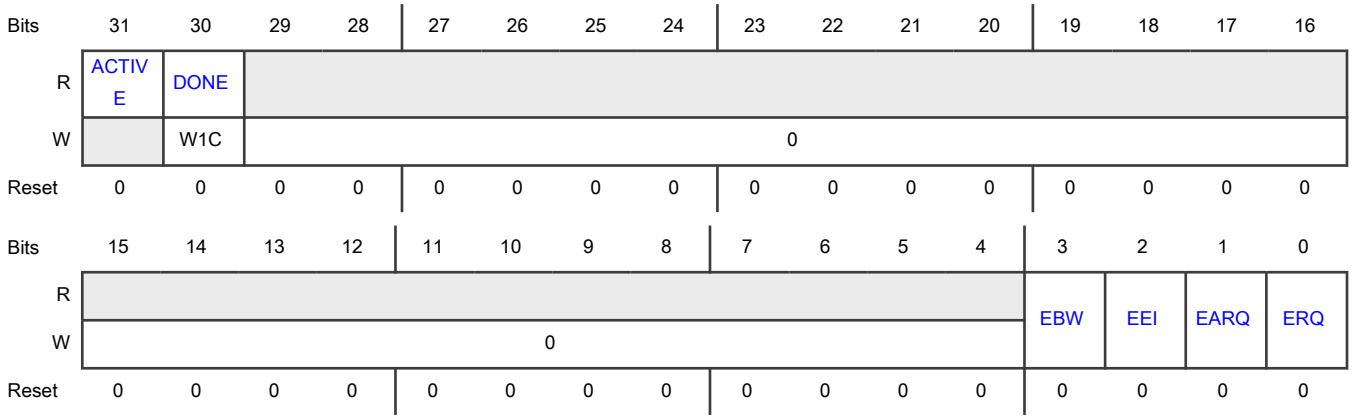
For n = 0 to 15:

Register	Offset
CHn_CSR	0h + (n × 1000h)

Function

This register contains several fields related to hardware and interrupt requests, configuration, and status for the given channel.

Diagram



Fields

Field	Function
31 ACTIVE	Channel Active The ACTIVE field indicates the channel was selected by arbitration and is executing the prescribed transfers. The eDMA sets it to 1 when channel service begins, and clears it to 0 as the minor loop completes or when any error condition is detected. Except for dynamic scatter/gather or dynamic channel linking, you must not modify the transfer control descriptor when a channel is active.
30 DONE	Channel Done The DONE field indicates the eDMA has completed the major loop. The eDMA engine sets this field as the CITER count reaches zero. If enabled, the eDMA generates an interrupt request corresponding to this completed channel. The software clears it, or the hardware clears it when the channel is activated. **NOTE** This field must be cleared to 0 before writing the MAJORELINK or ESG fields.
29-4 —	Reserved
3 EBW	Enable Buffered Writes When buffered writes are enabled, all writes except for the last write sequence of the minor loop are signaled by the eDMA as bufferable. 0b - Buffered writes on system bus disabled. Buffered writes on system bus disabled 1b - Buffered writes on system bus enabled. Bufferable write signal asserted on all system bus writes except during last write sequence
2 EEI	Enable Error Interrupt The EEI field enables the error interrupt signal for the channel. The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller. 0b - Error signal for corresponding channel does not generate error interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Assertion of error signal for corresponding channel generates error interrupt request
1 EARQ	Enable Asynchronous DMA Request The enable asynchronous DMA request field (EARQ) does not affect DMA operations. When set to 1, this field allows the hardware service request enable field (ERQ) to propagate out of the DMA to the power controller. When cleared to 0, this field masks the hardware service request enable field to the power controller. 0b - Disable asynchronous DMA request for the channel 1b - Enable asynchronous DMA request for the channel
0 ERQ	Enable DMA Request Disable a channel's hardware service request at the source before clearing the channel's ERQ field. The DMA hardware request input signal and the enable request field (ERQ) must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request field does not affect a channel service request made explicitly through software or channel linking. The state of the ERQ field does not affect the channel's START field. 0b - DMA hardware request signal for corresponding channel disabled 1b - DMA hardware request signal for corresponding channel enabled

23.6.3.3 Channel Error Status (CH0_ES - CH15_ES)

Offset

For n = 0 to 15:

Register	Offset
CHn_ES	4h + (n × 1000h)

Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- An illegal setting in the transfer control descriptor
- An error termination to a bus master read or write cycle

The ERR field signals the presence of an error for the channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate field in this register. The outputs of this register are enabled by the contents of the CHn_CSR[EEI] field, then logically summed across all channels to form an error interrupt request, which may be routed to the interrupt controller. In addition, this enabled error status is logically OR'd onto the channel done interrupt, CHn_INT[INT], thus forming a done or error interrupt on a per channel basis.

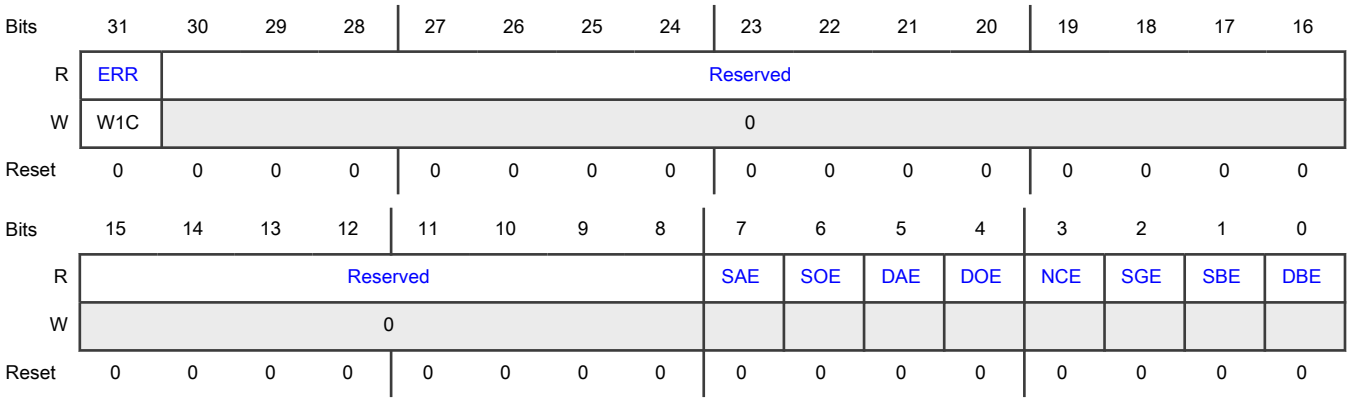
During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when eDMA detects an error. The contents of this ERR register field can also be polled because a non-zero value indicates the presence of a channel error, regardless of the state of the EEI mask.

The state of any given channel's error indicators is affected by writes to this register. Writing a 1 to the ERR field clears the channel's error status, and writing a 0 has no effect.

An unspecified error, where only the ERR field is set to 1, indicates that either a transfer was cancelled with an error. The Management Page Error Status register has full view of the error condition.

See Fault reporting and handling for more details.

Diagram



Fields

Field	Function
31 ERR	Error In Channel 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
30-8 —	Reserved
7 SAE	Source Address Error TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SADDR field
6 SOE	Source Offset Error TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SOFF field
5 DAE	Destination Address Error TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DADDR field

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 DOE	Destination Offset Error TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DOFF field
3 NCE	NBYTES/CITER Configuration Error This error indicates that one of the following has occurred: • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE] • TCDn_CITER[CITER] is equal to zero • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK] 0b - No NBYTES/CITER configuration error 1b - Last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields
2 SGE	Scatter/Gather Configuration Error When this field is 1, it indicates that TCDn_DLAST_SGA is not on a 32-byte boundary. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG]is enabled. 0b - No scatter/gather configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DLAST_SGA field
1 SBE	Source Bus Error 0b - No source bus error 1b - Last recorded error was bus error on source read
0 DBE	Destination Bus Error 0b - No destination bus error 1b - Last recorded error was bus error on destination write

23.6.3.4 Channel Interrupt Status (CH0_INT - CH15_INT)

Offset

For n = 0 to 15:

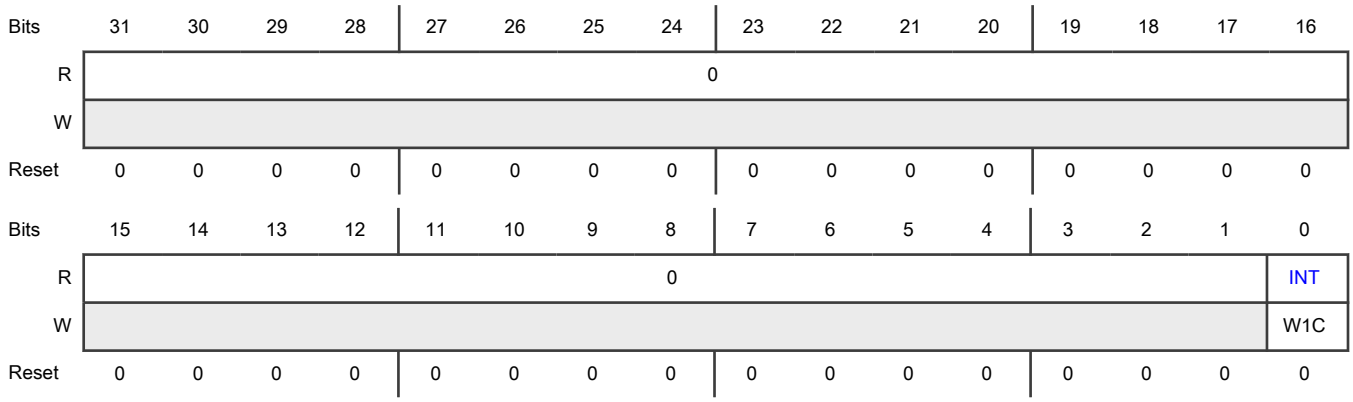
Register	Offset
CHn_INT	8h + (n × 1000h)

Function

The INT field signals the presence of an interrupt request for the channel. Depending on the appropriate bit setting in the transfer control descriptors, the eDMA engine generates an interrupt on data transfer completion or an error condition.

The outputs of this register are directly routed to the interrupt controller. During the interrupt service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. On writes to INT, a 1 clears the channel's interrupt request. A zero has no effect on the channel's current interrupt status.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 INT	Interrupt Request 0b - Interrupt request for corresponding channel cleared 1b - Interrupt request for corresponding channel active

23.6.3.5 Channel System Bus (CH0_SBR - CH15_SBR)

Offset

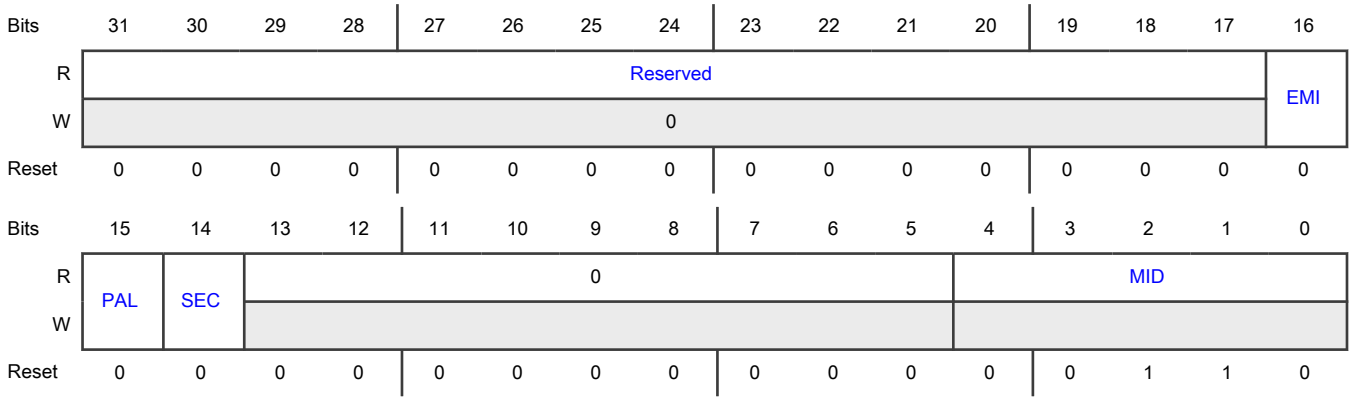
For n = 0 to 15:

Register	Offset
CHn_SBR	Ch + (n × 1000h)

Function

The Channel System Bus register places identification and attribute information on the system bus interface for the eDMA.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 EMI	Enable Master ID Replication The eDMA master ID replication field allows the eDMA to use the same protection level and system bus ID of the master programming the eDMA's TCD. When enabled, the eDMA uses the master ID and protection level stored in the CHn_SBR registers, instead of the eDMA's default values. When a master (for example a core) programs a TCD, its master ID is captured when the TCD*n*_CSR control attributes are written. A scatter/gather operation does not affect the CHn_SBR registers. You can write the EMI only if MP_CSR[GMRC] = 1, which means Global Master ID Replication Control is enabled; otherwise, the EMI is forced to zero. **NOTE** If master ID replication is disabled, the nonsecure, user protection level for DMA transfers is used. 0b - Master ID replication is disabled 1b - Master ID replication is enabled
15 PAL	Privileged Access Level This field controls DMA's protection level on the system bus when the channel is active. **NOTE** The value written into this register cannot exceed the security and privilege level of the core or other master writing the channel's system bus register; CH*n*_SBR. The order of precedence is SecurePriv>SecureUser>NonsecurePriv>NonsecureUser 0b - User protection level for DMA transfers 1b - Privileged protection level for DMA transfers
14 SEC	Security Level DMA's security level on the system bus when the channel is active.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The value written into this register cannot exceed the security and privilege level of the core or other master writing the channel's system bus register; CH*n*_SBR. The order of precedence is SecurePriv>SecureUser>NonsecurePriv>NonsecureUser 0b - Nonsecure protection level for DMA transfers 1b - Secure protection level for DMA transfers
13-5 —	Reserved
4-0 MID	Master ID This field controls the DMA's master ID on the system bus when the channel is active. **NOTE** The ID captured in this register reflects the master ID of the core or other master writing the channel's security attributes, TCD*n*_SBR[SEC].

23.6.3.6 Channel Priority (CH0_PRI - CH15_PRI)

Offset

For n = 0 to 15:

Register	Offset
CHn_PRI	10h + (n × 1000h)

Function

The contents of these registers define unique priorities associated with each channel within the same channel group. Channel grouping is programmed via Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI).

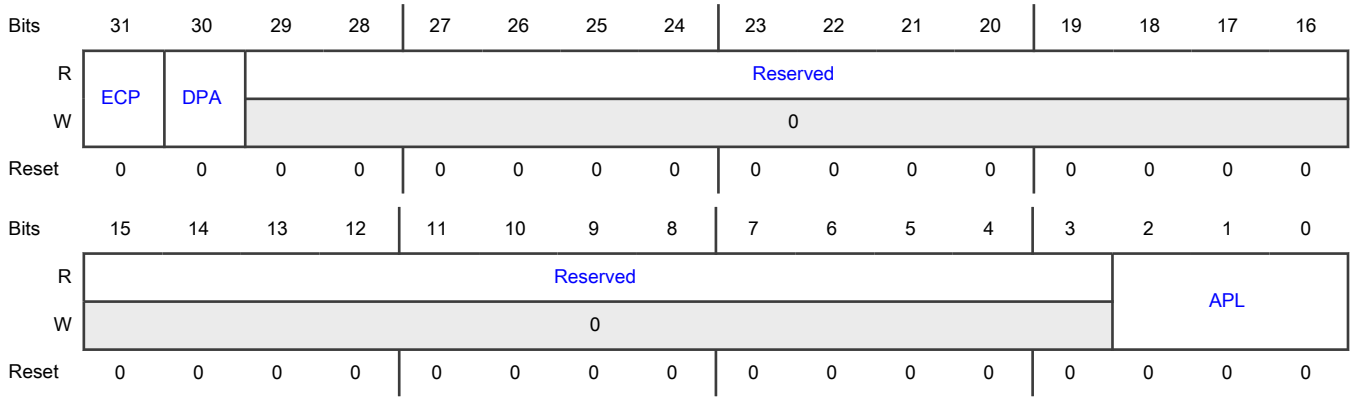
The channel priorities within a group are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values; otherwise, channel numbers with the same, non-zero value, will be selected based on channel number with the higher channel number having higher priority.

If more than one channel in a group has an arbitration priority level value of zero, then the arbitration mode field MP_CSR[ERCA] is used to determine the arbitration scheme for all channels with APL=0 within a group.

When you enable round-robin channel arbitration (MP_CSR[ERCA] = 1), all channels with APL=0 within a group will use a round-robin arbitration scheme, which rotates among these channels requesting service without regard to priority. Round-robin provides a fairness mechanism within an arbitration group.

When you enable fixed-priority channel arbitration (MP_CSR[ERCA] = 0), eDMA selects channels with APL=0 based on channel number, with the higher channel number having higher priority.

Diagram



Fields

Field	Function
31 ECP	Enable Channel Preemption 0b - Channel cannot be suspended by a higher-priority channel's service request 1b - Channel can be temporarily suspended by a higher-priority channel's service request
30 DPA	Disable Preempt Ability 0b - Channel can suspend a lower-priority channel 1b - Channel cannot suspend any other channel, regardless of channel priority
29-3 —	Reserved
2-0 APL	Arbitration Priority Level Channel priority level for arbitration within the assigned arbitration group.

23.6.3.7 Channel Multiplexor Configuration (CH0_MUX - CH15_MUX)

Offset

For n = 0 to 15:

Register	Offset
CHn_MUX	14h + (n × 1000h)

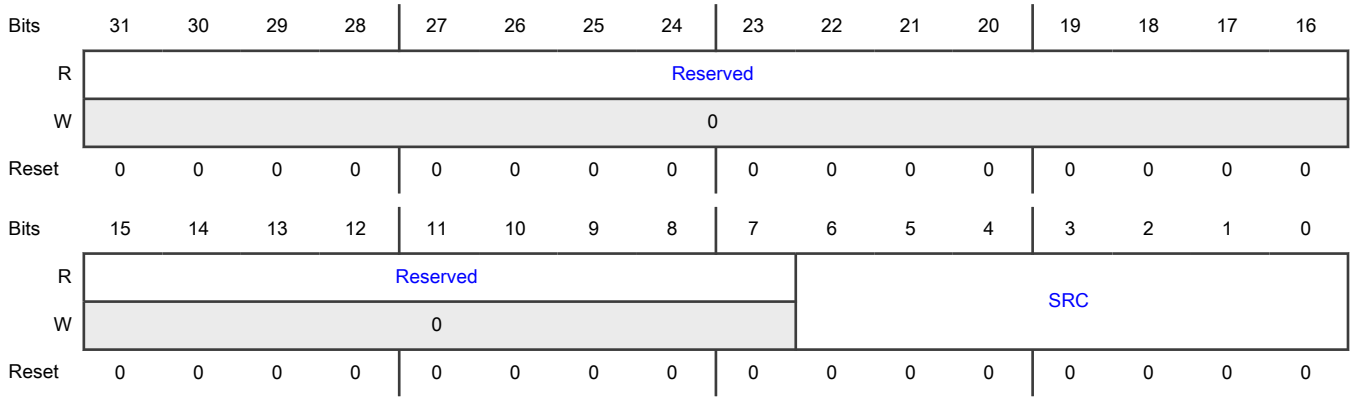
Function

Each of the DMA channels can be independently associated with various peripherals in the system. The Channel Multiplexor Configuration register selects the peripheral assigned to each channel. Service requests from the peripheral should be disabled when configuring a channel to a peripheral source.

Each channel must have a unique value when selecting a peripheral slot in the channel mux configuration. The only value that may overlap is source 0. If there is an attempt to write a mux configuration value that is already consumed by any channel, a mux configuration of 0 (SRC = 0) will be written.

All channels will default to source 0. When a particular peripheral is needed, the channel's mux configuration is set to that source number. When the peripheral is no longer needed, the mux configuration for that channel should be written to 0, thus releasing the resource.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 SRC	Service Request Source Hardware service request source for the channel. **NOTE** With the exception of 0, attempts to write a value already in use will be forced to 0.

23.6.3.8 TCD Source Address (TCD0_SADDR - TCD15_SADDR)

Offset

For n = 0 to 15:

Register	Offset
TCDn_SADDR	20h + (n × 1000h)

Function

This register contains the address for the read transactions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SADDR															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SADDR															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 SADDR	Source Address Memory address pointing to the source data.

23.6.3.9 TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)

Offset

For n = 0 to 15:

Register	Offset
TCDn_SOFF	24h + (n × 1000h)

Function

This register contains the sign-extended value added to Source Address register after each read transaction.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SOFF															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
15-0 SOFF	Source Address Signed Offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

23.6.3.10 TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)

Offset

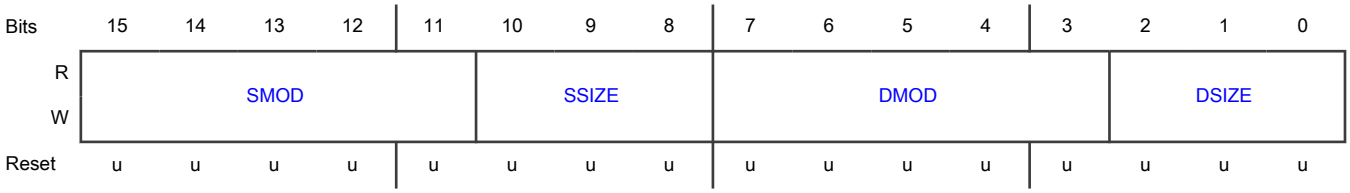
For n = 0 to 15:

Register	Offset
TCDn_ATTR	26h + (n × 1000h)

Function

This register contains size and option modulo addressing information for source and destination addresses.

Diagram



Fields

Field	Function
15-11 SMOD	Source Address Modulo This field defines a specific address range, which is the value after the SADDR + SOFF calculation is performed on the original register value. Setting this field makes it easy to implement a circular data queue. For data queues requiring power-of-2-sized bytes, the queue must start at a 0-modulo-size address and the SMOD field must be set to the appropriate value for the queue, freezing the required number of upper address bits. The value programmed into this field specifies the number of lower address bits that are allowed to change. For a circular queue application, you typically set TCDn_SOFF[SOFF] to the transfer size to implement post-increment addressing, with the SMOD function constraining the addresses to a 0-modulo-size range. 0_0000b - Source address modulo feature disabled 0_0001b - Source address modulo feature enabled for any non-zero value [1-31]
10-8 SSIZE	Source Data Transfer Size 000b - 8-bit 001b - 16-bit 010b - 32-bit 011b - 64-bit 100b - 16-byte 101b - 32-byte 110b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	111b - Reserved
7-3 DMOD	Destination Address Modulo See the SMOD definition.
2-0 DSIZE	Destination Data Transfer Size See the SSIZE definition.

23.6.3.11 TCD Transfer Size Without Minor Loop Offsets (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)

Offset

For n = 0 to 15:

Register	Offset
TCDn_NBYTES_MLOFFNO	28h + (n × 1000h)

Function

The TCDn_NBYTES field defines the number of bytes to transfer per service request.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR), or destination address (TCDn_DADDR), upon minor loop completion. Minor loop completion is when the channel has finished the service request and has transferred NBYTES. When minor loop offsets are enabled, the minor loop offset value (TCDn_NBYTES_MLOFFYES[MLOFF]) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both, prior to the addresses being written back to the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST_SDA and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (SMLOE or DMLOE is 1), TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is redefined. A portion of TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is used to specify multiple fields:

- A source enable bit (SMLOE) to specify the minor loop offset must be applied to the source address (TCDn_SADDR) upon minor loop completion
- A destination enable bit (DMLOE) to specify the minor loop offset must be applied to the destination address (TCDn_DADDR) upon minor loop completion
- The sign extended minor loop offset value (MLOFF)

The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. If both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

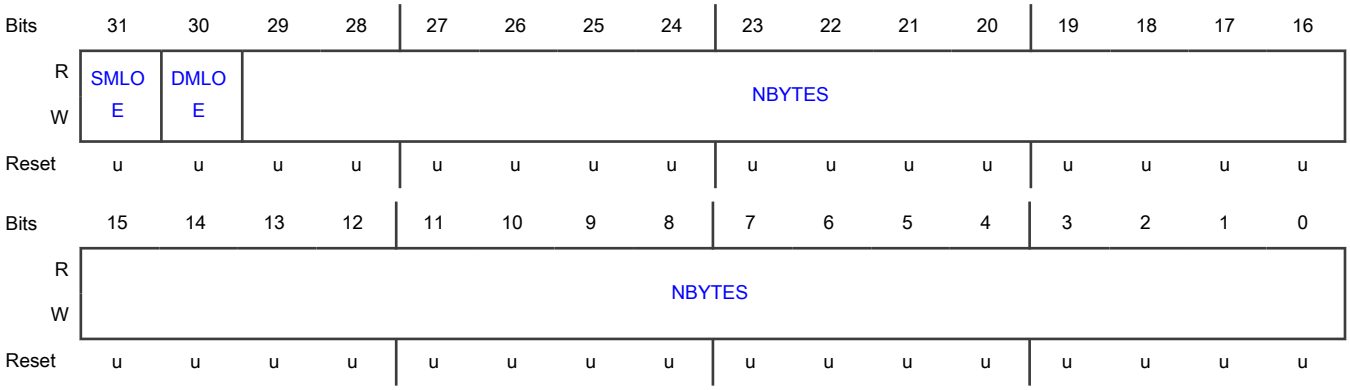
One of two register profiles (this register or TCDn_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether source or destination minor loop mapping is enabled.

TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is defined as follows:

- If SMLOE = 0 and DMLOE = 0, then see the TCDn_NBYTES_MLOFFNO register description.

- If either SMLOE or DMLOE is 1, then see the TCDn_NBYTES_MLOFFYES register description.

Diagram



Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0b - Minor loop offset not applied to SADDR 1b - Minor loop offset applied to SADDR
30 DMLOE	Destination Minor Loop Offset Enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0b - Minor loop offset not applied to DADDR 1b - Minor loop offset applied to DADDR
29-0 NBYTES	Number of Bytes To Transfer Per Service Request Number of bytes to be transferred for each service request of the channel. When a channel activates, the module loads the appropriate TCD contents into the eDMA engine and performs the appropriate reads and writes until the byte transfer count has been reached. This process is normally an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the byte count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major loop iteration count (CITER) is decremented by one and written back to the TCD memory. If the major iteration count is complete, additional processing is performed.

23.6.3.12 TCD Transfer Size with Minor Loop Offsets (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)

Offset

For n = 0 to 15:

Register	Offset
TCDn_NBYTES_MLOFFYES	28h + (n × 1000h)

Function

The TCDn_NBYTES field defines the number of bytes to transfer per service request.

Minor loop offset is an address offset value added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. Minor loop completion occurs when the channel has finished the service request and has transferred NBYTES. Minor loop offsets are enabled by setting either the source enable bit (SMLOE) or the destination enable bit (DMLOE).

The source enable bit (SMLOE) specifies the minor loop offset value (MLOFF) that is to be applied to the source address (TCDn_SADDR) upon minor loop completion. The destination enable bit (DMLOE) specifies the minor loop offset (MLOFF) that is to be applied to the destination address (TCDn_DADDR) upon minor loop completion.

If the major loop is complete, the minor loop offsets are ignored and the major loop address offsets (TCDn_SLAST_SDA and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

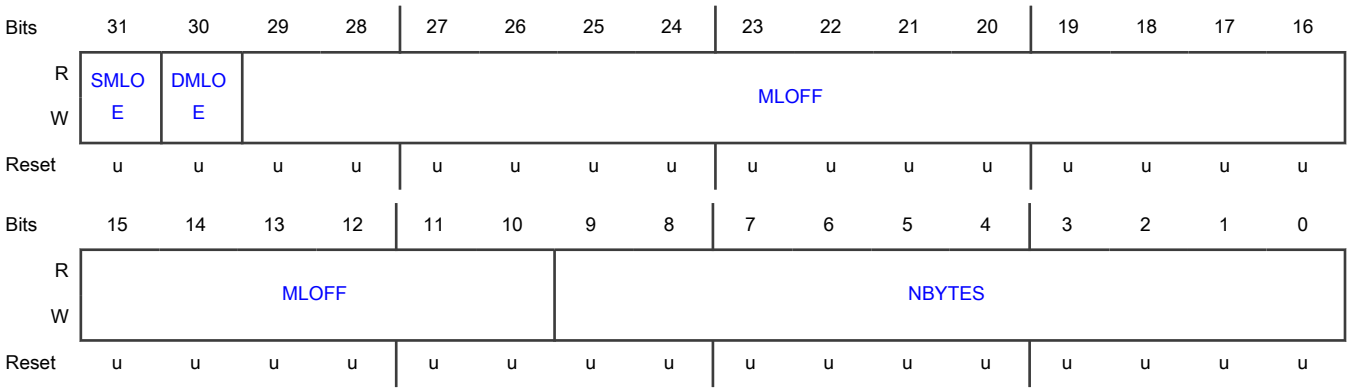
When you enable the minor loop offset overlay (either SMLOE or DMLOE is 1), eDMA redefines TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES. A portion of TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES specifies the sign-extended minor loop offset value (MLOFF). The same offset value (MLOFF) applies to both source and destination minor loop offsets. When the minor loop offset is enabled, you must align it to the transfer size of the source or destination it is associated with. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. If both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

One of two register profiles (this register or TCDn_NBYTES_MLOFFNO) defines the number of bytes to transfer per request. Which register to use depends on whether source or destination minor loop mapping is enabled.

TCDn_NBYTES_MLOFFYES is defined as follows:

- If either minor loop offset is enabled (SMLOE or DMLOE = 1), then see the TCDn_NBYTES_MLOFFYES register description.
- If SMLOE and DMLOE are both 0, then see the TCDn_NBYTES_MLOFFNO register description.

Diagram



Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0b - Minor loop offset not applied to SADDR 1b - Minor loop offset applied to SADDR
30 DMLOE	Destination Minor Loop Offset Enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0b - Minor loop offset not applied to DADDR 1b - Minor loop offset applied to DADDR
29-10 MLOFF	Minor Loop Offset If SMLOE or DMLOE is 1, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
9-0 NBYTES	Number of Bytes To Transfer Per Service Request The number of bytes to be transferred in each service request of the channel. As a channel activates, the module loads the appropriate TCD contents into the eDMA engine and performs the appropriate reads and writes until the minor byte transfer count has been reached. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is complete, additional processing is performed.

23.6.3.13 TCD Last Source Address Adjustment / Store DADDR Address (TCD0_SLAST_SDA - TCD15_SLAST_SDA)

Offset

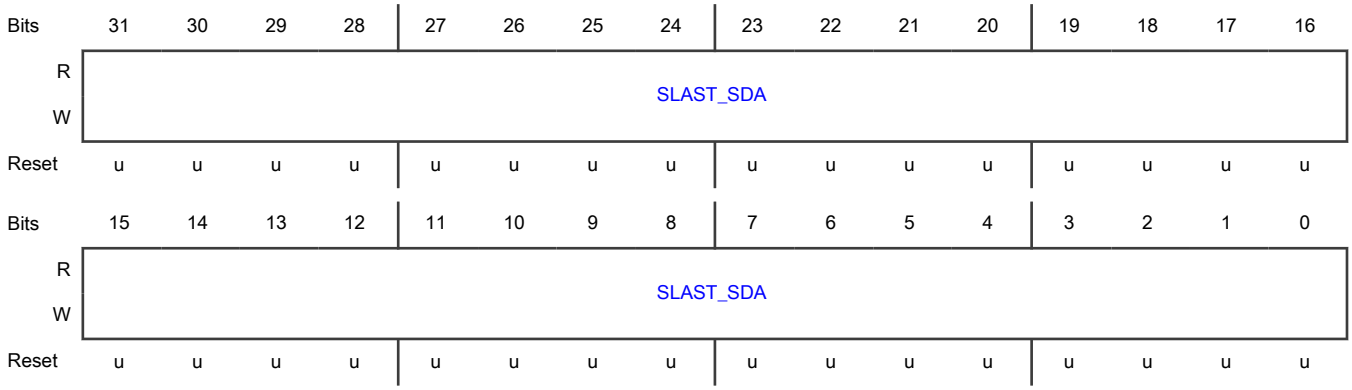
For n = 0 to 15:

Register	Offset
TCDn_SLAST_SDA	2Ch + (n × 1000h)

Function

This register contains the value added to the source address when the major loop is complete. When the store destination address option is enabled, this field provides a pointer to memory for storing the final destination address.

Diagram



Fields

Field	Function
31-0 SLAST_SDA	Last Source Address Adjustment / Store DADDR Address Source last address adjustment or the system memory address for destination address (DADDR) storage. If (TCDn_CSR[ESDA] = 0), then: • Adjustment value is added to the source address at the completion of the major iteration count. This value can be used to restore the source address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final source address adjustment. Otherwise: • This address points to the 32-bit-aligned memory location where the destination address (DADDR) is to be stored in system memory. By saving the final destination address in system memory via the ESDA feature, you are able to compute the size of a variable destination data buffer by simply subtracting the beginning DADDR from the final, saved DADDR. This feature is used together with the scatter/gather operation to prevent the loss of the final DADDR, which is overwritten during the scatter/gather operation. The "Store Destination Address" (SDA) value must be a 32-bit-aligned location because the eDMA forces the lower two address bits of the SLAST_SDA field to zero when ESDA is enabled. The module performs this write operation when the major loop is done; that is, when the major iteration count (CITER) decrements to zero.

23.6.3.14 TCD Destination Address (TCD0_DADDR - TCD15_DADDR)

Offset

For n = 0 to 15:

Register	Offset
TCDn_DADDR	30h + (n × 1000h)

Function

This register contains the address for the write transactions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DADDR															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DADDR															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 DADDR	Destination Address Memory address pointing to the destination data.

23.6.3.15 TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)

Offset

For n = 0 to 15:

Register	Offset
TCDn_DOFF	34h + (n × 1000h)

Function

This register contains the sign-extended value added to Destination Address register after each write transaction.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DOFF															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
15-0 DOFF	Destination Address Signed Offset Sign-extended offset that is applied to the current destination address to form the next-state value as each destination write is completed.

23.6.3.16 TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)

Offset

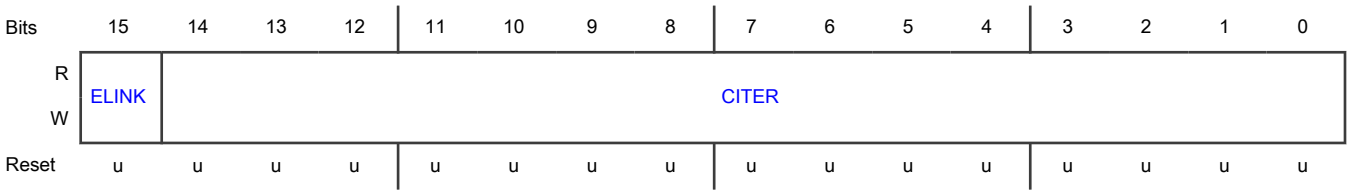
For n = 0 to 15:

Register	Offset
TCDn_CITER_ELINKNO	36h + (n × 1000h)

Function

If TCDn_CITER[ELINK] is 0, the TCDn_CITER register is defined as follows.

Diagram



Fields

Field	Function
15 ELINK	Enable Link As the channel completes the minor loop, this flag enables linking to another channel as defined by the relevant LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel to 1. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of MAJORELINK channel linking. **NOTE** This field must be equal to the BITER[ELINK] field; otherwise, a configuration error is reported. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-0 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the channel finishes a service request and is written back to TCD memory. After the major iteration count is exhausted, the channel performs a number of operations — for example, final source and destination address calculations — and optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.6.3.17 TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)

Offset

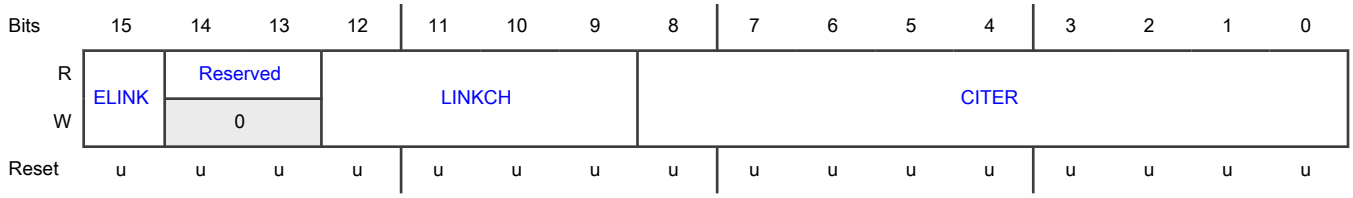
For n = 0 to 15:

Register	Offset
TCDn_CITER_ELINKYES	36h + (n × 1000h)

Function

If TCDn_CITER[ELINK] is 1, the TCDn_CITER register is defined as follows.

Diagram



Fields

Field	Function
15 ELINK	Enable Link As the channel completes the minor loop, this flag enables linking to another channel as defined by the relevant LINKCH field. When enabled, an internal mechanism sets the TCDn_CSR[START] field of the specified channel (LINKCH) upon minor loop completion. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of MAJORELINK channel linking.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field must be equal to the BITER[ELINK] field; otherwise, a configuration error is reported. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-13 —	Reserved
12-9 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted the eDMA engine initiates a channel service request to the channel defined by this field by writing that channel's TCDn_CSR[START] field to 1.
8-0 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the channel finishes a service request and is written back to the TCD memory. After the major iteration count is exhausted, the channel performs a number of operations — for example, final source and destination address calculations — and optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.6.3.18 TCD Last Destination Address Adjustment / Scatter Gather Address (TCD0_DLAST_SGA - TCD15_DLAST_SGA)

Offset

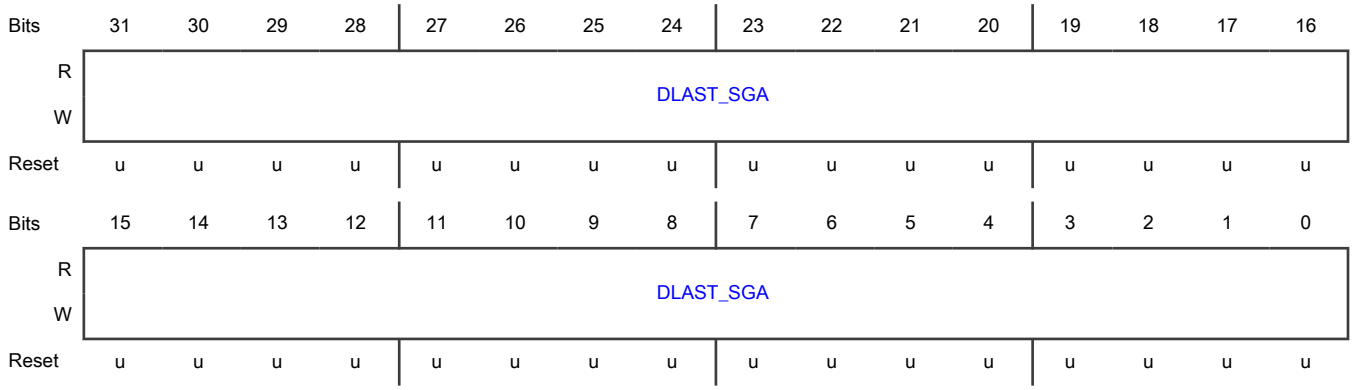
For n = 0 to 15:

Register	Offset
TCDn_DLAST_SGA	38h + (n × 1000h)

Function

This register contains the value added to the destination address when the major loop is complete. When the Scatter/Gather option is enabled, this field provides a pointer to memory for fetching a transfer control descriptor to reprogram the channel.

Diagram



Fields

Field	Function
31-0 DLAST_SGA	Last Destination Address Adjustment / Scatter Gather Address Adjustment of the last destination address or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • Adjustment value is added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo 32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo 32-byte, or else a configuration error is reported.

23.6.3.19 TCD Control and Status (TCD0_CSR - TCD15_CSR)

Offset

For n = 0 to 15:

Register	Offset
TCDn_CSR	3Ch + (n × 1000h)

Function

This register is used to enable optional features.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWC				MAJORLINKCH				ESDA	EEOP	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
W			0													
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	0

Fields

Field	Function
15-14 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces eDMA to stall after the completion of each read/write access, to control the bus request bandwidth seen by the system bus interconnect. **NOTE** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00b - No eDMA engine stalls 01b - Reserved 10b - eDMA engine stalls for 4 cycles after each R/W 11b - eDMA engine stalls for 8 cycles after each R/W
13-12 —	Reserved
11-8 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] field to 1.
7 ESDA	Enable Store Destination Address As the channel completes the major loop by either the current iteration counter (CITER) decrementing to 0, or by receiving an enabled end-of-packet signal, this field enables writing the destination address (DADDR) to the address stored in the SLAST_SDA field. The value written to system memory is the last DADDR value prior to the DLAST_SGA offset being applied, or overwritten by an enabled scatter/gather operation. When the ESDA bit is 1, SLAST_SDA contains the write pointer instead of the final source address offset. Because this is a pointer and not a final offset, a last source address offset of zero is applied to SADDR instead of the SLAST_SGA value. 0b - Ability to store destination address to system memory disabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Ability to store destination address to system memory enabled
6 EEOP	Enable End-Of-Packet Processing When enabled by the EEOP field, an end-of-packet hardware input signal directs eDMA to discontinue executing the active channel, and to treat the shutdown as the major-loop-completed event. If the EEOP field is 1, the end-of-packet signal from supported peripherals is accepted. If the EEOP field is 0, the end-of-packet input is ignored. With an end-of-packet retirement, the current TCD destination address (or ESDA-saved destination address), minus the software-saved initial address (DADDR), reflects the total amount of data transferred. 0b - End-of-packet operation disabled 1b - End-of-packet hardware input signal enabled
5 MAJORELINK	Enable Link When Major Loop Complete As the channel completes the major loop, this flag enables linking to another channel defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. **NOTE** To support the dynamic linking coherency model, this field is forced to 0 if written when TCDn_CSR[DONE] is 1. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
4 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses TCDn_DLAST_SGA as a memory pointer to a 0-modulo 32-bit address containing a 32-byte data structure, which is loaded as the transfer control descriptor into local memory. **NOTE** To support the dynamic scatter/gather coherency model, this field is forced to 0 if written when TCDn_CSR[DONE] is 1. 0b - Current channel's TCD is normal format 1b - Current channel's TCD specifies scatter/gather format.
3 DREQ	Disable Request If this flag is 1, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches 0. 0b - No operation. Channel's ERQ field not affected 1b - Clear the ERQ field to 0 upon major loop completion, thus disabling hardware service requests. Channel's ERQ field cleared to 0 when major loop complete
2	Enable Interrupt If Major Counter Half-complete

Table continues on the next page...

Table continued from the previous page...

Field	Function
INTHALF	If this flag is 1, the channel generates an interrupt request by setting the appropriate field in the INT register to 1 when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER = (BITER/2)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes, or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE** If BITER = 1, do not use INTHALF; use INTMAJOR instead. 0b - Halfway point interrupt disabled 1b - Halfway point interrupt enabled
1 INTMAJOR	Enable Interrupt If Major count complete If this flag is 1, the channel generates an interrupt request by setting the appropriate field in the INT register to 1 when the current major iteration count (CITER) reaches 0. 0b - End-of-major loop interrupt disabled 1b - End-of-major loop interrupt enabled
0 START	Channel Start If this flag is 1, the channel is requesting service. The eDMA hardware automatically clears this flag to 0 after the channel begins execution. 0b - Channel not explicitly started 1b - Channel explicitly started via a software-initiated service request

23.6.3.20 TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)

Offset

For n = 0 to 15:

Register	Offset
TCDn_BITER_ELINKNO	3Eh + (n × 1000h)

Function

If the TCDn_BITER[ELINK] field is 0, the TCDn_BITER register is defined as follows.

Diagram

Fields

Field	Function
15 ELINK	Enables Link As the channel completes the minor loop, this flag enables linking to another channel as defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-0 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be set equal to the value in the CITER field. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER must be 0x0001.

23.6.3.21 TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)

Offset

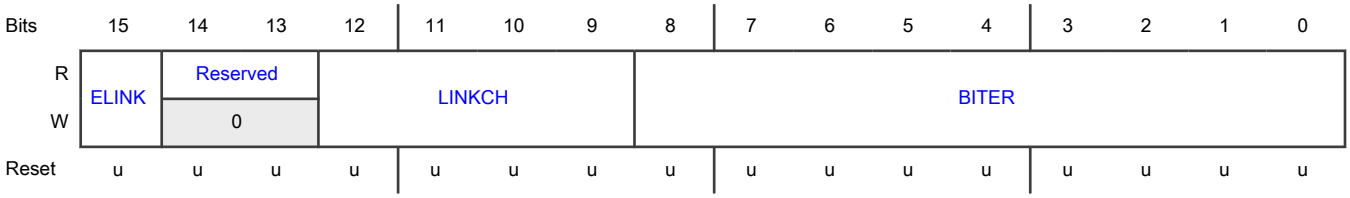
For n = 0 to 15:

Register	Offset
TCDn_BITER_ELINKYES	3Eh + (n × 1000h)

Function

If the TCDn_BITER[ELINK] field is set, the TCDn_BITER register is defined as follows.

Diagram



Fields

Field	Function
15 ELINK	Enable Link As the channel completes the minor loop, this flag enables linking to another channel as defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-13 —	Reserved
12-9 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] field. **NOTE** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field.
8-0 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be set equal to the value in the CITER field. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER must be 0x0001.

23.6.4 DMA TCD register descriptions

23.6.4.1 TCD memory map

eDMA_1_TCD base address: 400A_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h - F000h	Channel Control and Status (CH0_CSR - CH15_CSR)	32	RW	0000_0000h
4h - F004h	Channel Error Status (CH0_ES - CH15_ES)	32	RW	0000_0000h
8h - F008h	Channel Interrupt Status (CH0_INT - CH15_INT)	32	RW	0000_0000h
Ch - F00Ch	Channel System Bus (CH0_SBR - CH15_SBR)	32	RW	0000_0007h
10h - F010h	Channel Priority (CH0_PRI - CH15_PRI)	32	RW	0000_0000h
14h - F014h	Channel Multiplexor Configuration (CH0_MUX - CH15_MUX)	32	RW	0000_0000h
20h - F020h	TCD Source Address (TCD0_SADDR - TCD15_SADDR)	32	RW	See section
24h - F024h	TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)	16	RW	See section
26h - F026h	TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)	16	RW	See section
28h - F028h	TCD Transfer Size Without Minor Loop Offsets (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)	32	RW	See section
28h - F028h	TCD Transfer Size with Minor Loop Offsets (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)	32	RW	See section
2Ch - F02Ch	TCD Last Source Address Adjustment / Store DADDR Address (TCD0_SLAST_SDA - TCD15_SLAST_SDA)	32	RW	See section
30h - F030h	TCD Destination Address (TCD0_DADDR - TCD15_DADDR)	32	RW	See section
34h - F034h	TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)	16	RW	See section
36h - F036h	TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)	16	RW	See section
36h - F036h	TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)	16	RW	See section
38h - F038h	TCD Last Destination Address Adjustment / Scatter Gather Address (TCD0_DLAST_SGA - TCD15_DLAST_SGA)	32	RW	See section
3Ch - F03Ch	TCD Control and Status (TCD0_CSR - TCD15_CSR)	16	RW	See section
3Eh - F03Eh	TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)	16	RW	See section
3Eh - F03Eh	TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)	16	RW	See section

23.6.4.2 Channel Control and Status (CH0_CSR - CH15_CSR)

Offset

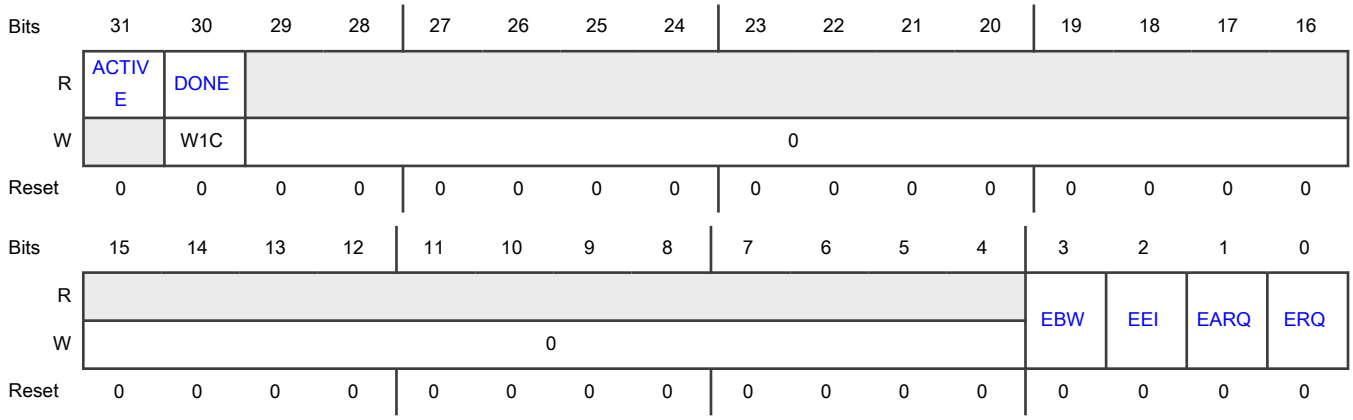
For n = 0 to 15:

Register	Offset
CHn_CSR	0h + (n × 1000h)

Function

This register contains several fields related to hardware and interrupt requests, configuration, and status for the given channel.

Diagram



Fields

Field	Function
31 ACTIVE	Channel Active The ACTIVE field indicates the channel was selected by arbitration and is executing the prescribed transfers. The eDMA sets it to 1 when channel service begins, and clears it to 0 as the minor loop completes or when any error condition is detected. Except for dynamic scatter/gather or dynamic channel linking, you must not modify the transfer control descriptor when a channel is active.
30 DONE	Channel Done The DONE field indicates the eDMA has completed the major loop. The eDMA engine sets this field as the CITER count reaches zero. If enabled, the eDMA generates an interrupt request corresponding to this completed channel. The software clears it, or the hardware clears it when the channel is activated. **NOTE** This field must be cleared to 0 before writing the MAJORELINK or ESG fields.
29-4 —	Reserved
3 EBW	Enable Buffered Writes When buffered writes are enabled, all writes except for the last write sequence of the minor loop are signaled by the eDMA as bufferable. 0b - Buffered writes on system bus disabled. Buffered writes on system bus disabled 1b - Buffered writes on system bus enabled. Bufferable write signal asserted on all system bus writes except during last write sequence
2 EEI	Enable Error Interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The EEI field enables the error interrupt signal for the channel. The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller. 0b - Error signal for corresponding channel does not generate error interrupt 1b - Assertion of error signal for corresponding channel generates error interrupt request
1 EARQ	Enable Asynchronous DMA Request The enable asynchronous DMA request field (EARQ) does not affect DMA operations. When set to 1, this field allows the hardware service request enable field (ERQ) to propagate out of the DMA to the power controller. When cleared to 0, this field masks the hardware service request enable field to the power controller. 0b - Disable asynchronous DMA request for the channel 1b - Enable asynchronous DMA request for the channel
0 ERQ	Enable DMA Request Disable a channel's hardware service request at the source before clearing the channel's ERQ field. The DMA hardware request input signal and the enable request field (ERQ) must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request field does not affect a channel service request made explicitly through software or channel linking. The state of the ERQ field does not affect the channel's START field. 0b - DMA hardware request signal for corresponding channel disabled 1b - DMA hardware request signal for corresponding channel enabled

23.6.4.3 Channel Error Status (CH0_ES - CH15_ES)

Offset

For n = 0 to 15:

Register	Offset
CHn_ES	4h + (n × 1000h)

Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- An illegal setting in the transfer control descriptor
- An error termination to a bus master read or write cycle

The ERR field signals the presence of an error for the channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate field in this register. The outputs of this register are enabled by the contents of the CHn_CSR[EEI] field, then logically summed across all channels to form an error interrupt request, which may be routed to the interrupt controller. In addition, this enabled error status is logically OR'd onto the channel done interrupt, CHn_INT[INT], thus forming a done or error interrupt on a per channel basis.

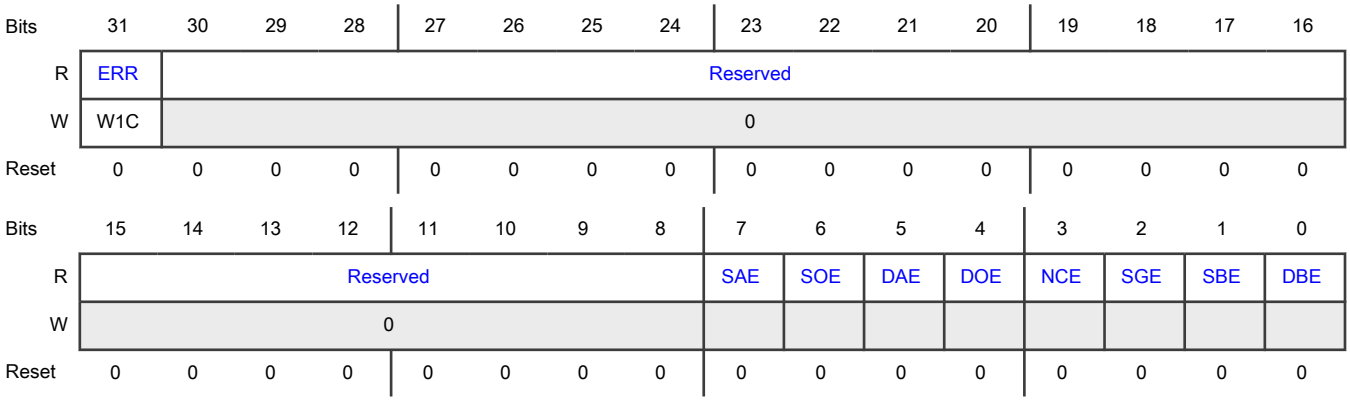
During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when eDMA detects an error. The contents of this ERR register field can also be polled because a non-zero value indicates the presence of a channel error, regardless of the state of the EEI mask.

The state of any given channel's error indicators is affected by writes to this register. Writing a 1 to the ERR field clears the channel's error status, and writing a 0 has no effect.

An unspecified error, where only the ERR field is set to 1, indicates that either a transfer was cancelled with an error. The Management Page Error Status register has full view of the error condition.

See Fault reporting and handling for more details.

Diagram



Fields

Field	Function
31 ERR	Error In Channel 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
30-8 —	Reserved
7 SAE	Source Address Error TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SADDR field
6 SOE	Source Offset Error TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE]. 0b - No source offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_SOFF field

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 DAE	Destination Address Error TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination address configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DADDR field
4 DOE	Destination Offset Error TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. 0b - No destination offset configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DOFF field
3 NCE	NBYTES/CITER Configuration Error This error indicates that one of the following has occurred: • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE] • TCDn_CITER[CITER] is equal to zero • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK] 0b - No NBYTES/CITER configuration error 1b - Last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields
2 SGE	Scatter/Gather Configuration Error When this field is 1, it indicates that TCDn_DLAST_SGA is not on a 32-byte boundary. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG]is enabled. 0b - No scatter/gather configuration error 1b - Last recorded error was a configuration error detected in the TCDn_DLAST_SGA field
1 SBE	Source Bus Error 0b - No source bus error 1b - Last recorded error was bus error on source read
0 DBE	Destination Bus Error 0b - No destination bus error 1b - Last recorded error was bus error on destination write

23.6.4.4 Channel Interrupt Status (CH0_INT - CH15_INT)

Offset

For n = 0 to 15:

Register	Offset
CHn_INT	8h + (n × 1000h)

Function

The INT field signals the presence of an interrupt request for the channel. Depending on the appropriate bit setting in the transfer control descriptors, the eDMA engine generates an interrupt on data transfer completion or an error condition.

The outputs of this register are directly routed to the interrupt controller. During the interrupt service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. On writes to INT, a 1 clears the channel's interrupt request. A zero has no effect on the channel's current interrupt status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															INT
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 INT	Interrupt Request 0b - Interrupt request for corresponding channel cleared 1b - Interrupt request for corresponding channel active

23.6.4.5 Channel System Bus (CH0_SBR - CH15_SBR)

Offset

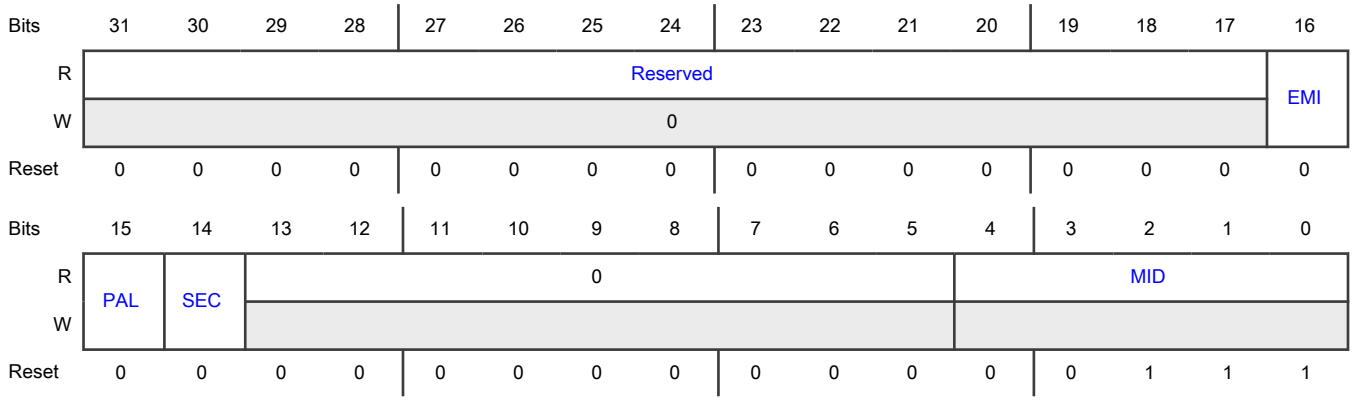
For n = 0 to 15:

Register	Offset
CHn_SBR	Ch + (n × 1000h)

Function

The Channel System Bus register places identification and attribute information on the system bus interface for the eDMA.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 EMI	Enable Master ID Replication The eDMA master ID replication field allows the eDMA to use the same protection level and system bus ID of the master programming the eDMA's TCD. When enabled, the eDMA uses the master ID and protection level stored in the CHn_SBR registers, instead of the eDMA's default values. When a master (for example a core) programs a TCD, its master ID is captured when the TCD*n*_CSR control attributes are written. A scatter/gather operation does not affect the CHn_SBR registers. You can write the EMI only if MP_CSR[GMRC] = 1, which means Global Master ID Replication Control is enabled; otherwise, the EMI is forced to zero. **NOTE** If master ID replication is disabled, the nonsecure, user protection level for DMA transfers is used. 0b - Master ID replication is disabled 1b - Master ID replication is enabled
15 PAL	Privileged Access Level This field controls DMA's protection level on the system bus when the channel is active. **NOTE** The value written into this register cannot exceed the security and privilege level of the core or other master writing the channel's system bus register; CH*n*_SBR. The order of precedence is SecurePriv>SecureUser>NonsecurePriv>NonsecureUser 0b - User protection level for DMA transfers 1b - Privileged protection level for DMA transfers
14 SEC	Security Level DMA's security level on the system bus when the channel is active.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The value written into this register cannot exceed the security and privilege level of the core or other master writing the channel's system bus register; CH*n*_SBR. The order of precedence is SecurePriv>SecureUser>NonsecurePriv>NonsecureUser 0b - Nonsecure protection level for DMA transfers 1b - Secure protection level for DMA transfers
13-5 —	Reserved
4-0 MID	Master ID This field controls the DMA's master ID on the system bus when the channel is active. **NOTE** The ID captured in this register reflects the master ID of the core or other master writing the channel's security attributes, TCD*n*_SBR[SEC].

23.6.4.6 Channel Priority (CH0_PRI - CH15_PRI)

Offset

For n = 0 to 15:

Register	Offset
CHn_PRI	10h + (n × 1000h)

Function

The contents of these registers define unique priorities associated with each channel within the same channel group. Channel grouping is programmed via Channel Arbitration Group (CH0_GRPRI - CH15_GRPRI).

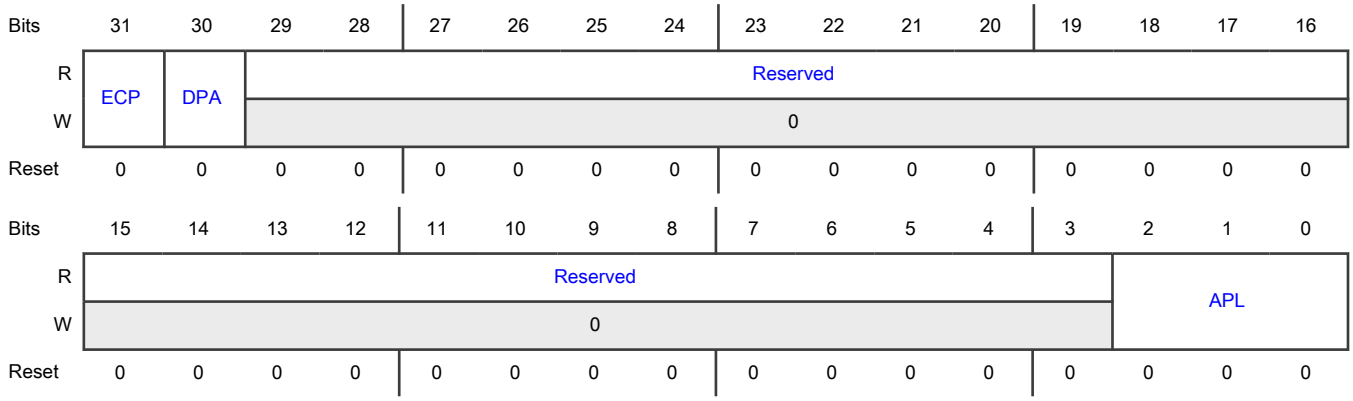
The channel priorities within a group are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values; otherwise, channel numbers with the same, non-zero value, will be selected based on channel number with the higher channel number having higher priority.

If more than one channel in a group has an arbitration priority level value of zero, then the arbitration mode field MP_CSR[ERCA] is used to determine the arbitration scheme for all channels with APL=0 within a group.

When you enable round-robin channel arbitration (MP_CSR[ERCA] = 1), all channels with APL=0 within a group will use a round-robin arbitration scheme, which rotates among these channels requesting service without regard to priority. Round-robin provides a fairness mechanism within an arbitration group.

When you enable fixed-priority channel arbitration (MP_CSR[ERCA] = 0), eDMA selects channels with APL=0 based on channel number, with the higher channel number having higher priority.

Diagram



Fields

Field	Function
31 ECP	Enable Channel Preemption 0b - Channel cannot be suspended by a higher-priority channel's service request 1b - Channel can be temporarily suspended by a higher-priority channel's service request
30 DPA	Disable Preempt Ability 0b - Channel can suspend a lower-priority channel 1b - Channel cannot suspend any other channel, regardless of channel priority
29-3 —	Reserved
2-0 APL	Arbitration Priority Level Channel priority level for arbitration within the assigned arbitration group.

23.6.4.7 Channel Multiplexor Configuration (CH0_MUX - CH15_MUX)

Offset

For n = 0 to 15:

Register	Offset
CHn_MUX	14h + (n × 1000h)

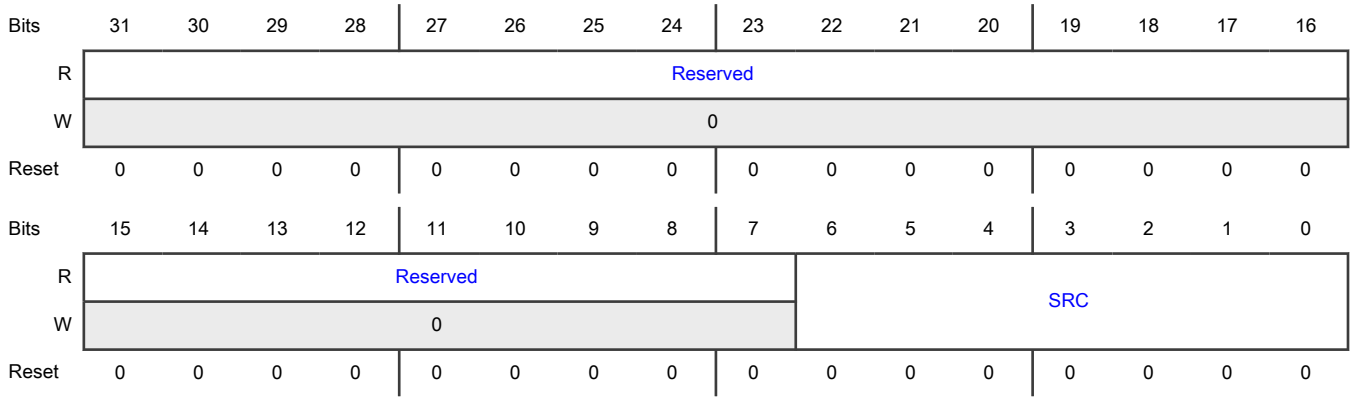
Function

Each of the DMA channels can be independently associated with various peripherals in the system. The Channel Multiplexor Configuration register selects the peripheral assigned to each channel. Service requests from the peripheral should be disabled when configuring a channel to a peripheral source.

Each channel must have a unique value when selecting a peripheral slot in the channel mux configuration. The only value that may overlap is source 0. If there is an attempt to write a mux configuration value that is already consumed by any channel, a mux configuration of 0 (SRC = 0) will be written.

All channels will default to source 0. When a particular peripheral is needed, the channel's mux configuration is set to that source number. When the peripheral is no longer needed, the mux configuration for that channel should be written to 0, thus releasing the resource.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 SRC	Service Request Source Hardware service request source for the channel. **NOTE** With the exception of 0, attempts to write a value already in use will be forced to 0.

23.6.4.8 TCD Source Address (TCD0_SADDR - TCD15_SADDR)

Offset

For n = 0 to 15:

Register	Offset
TCDn_SADDR	20h + (n × 1000h)

Function

This register contains the address for the read transactions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 SADDR	Source Address Memory address pointing to the source data.

23.6.4.9 TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)

Offset

For n = 0 to 15:

Register	Offset
TCDn_SOFF	24h + (n × 1000h)

Function

This register contains the sign-extended value added to Source Address register after each read transaction.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
15-0 SOFF	Source Address Signed Offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

23.6.4.10 TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)

Offset

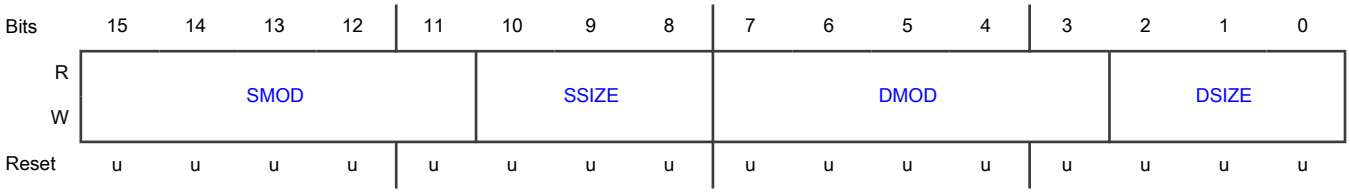
For n = 0 to 15:

Register	Offset
TCDn_ATTR	26h + (n × 1000h)

Function

This register contains size and option modulo addressing information for source and destination addresses.

Diagram



Fields

Field	Function
15-11 SMOD	Source Address Modulo This field defines a specific address range, which is the value after the SADDR + SOFF calculation is performed on the original register value. Setting this field makes it easy to implement a circular data queue. For data queues requiring power-of-2-sized bytes, the queue must start at a 0-modulo-size address and the SMOD field must be set to the appropriate value for the queue, freezing the required number of upper address bits. The value programmed into this field specifies the number of lower address bits that are allowed to change. For a circular queue application, you typically set TCDn_SOFF[SOFF] to the transfer size to implement post-increment addressing, with the SMOD function constraining the addresses to a 0-modulo-size range. 0_0000b - Source address modulo feature disabled 0_0001b - Source address modulo feature enabled for any non-zero value [1-31]
10-8 SSIZE	Source Data Transfer Size 000b - 8-bit 001b - 16-bit 010b - 32-bit 011b - 64-bit 100b - 16-byte 101b - 32-byte 110b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	111b - Reserved
7-3 DMOD	Destination Address Modulo See the SMOD definition.
2-0 DSIZE	Destination Data Transfer Size See the SSIZE definition.

23.6.4.11 TCD Transfer Size Without Minor Loop Offsets (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)

Offset

For n = 0 to 15:

Register	Offset
TCDn_NBYTES_MLOFFNO	28h + (n × 1000h)

Function

The TCDn_NBYTES field defines the number of bytes to transfer per service request.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR), or destination address (TCDn_DADDR), upon minor loop completion. Minor loop completion is when the channel has finished the service request and has transferred NBYTES. When minor loop offsets are enabled, the minor loop offset value (TCDn_NBYTES_MLOFFYES[MLOFF]) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both, prior to the addresses being written back to the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST_SDA and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (SMLOE or DMLOE is 1), TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is redefined. A portion of TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is used to specify multiple fields:

- A source enable bit (SMLOE) to specify the minor loop offset must be applied to the source address (TCDn_SADDR) upon minor loop completion
- A destination enable bit (DMLOE) to specify the minor loop offset must be applied to the destination address (TCDn_DADDR) upon minor loop completion
- The sign extended minor loop offset value (MLOFF)

The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. If both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

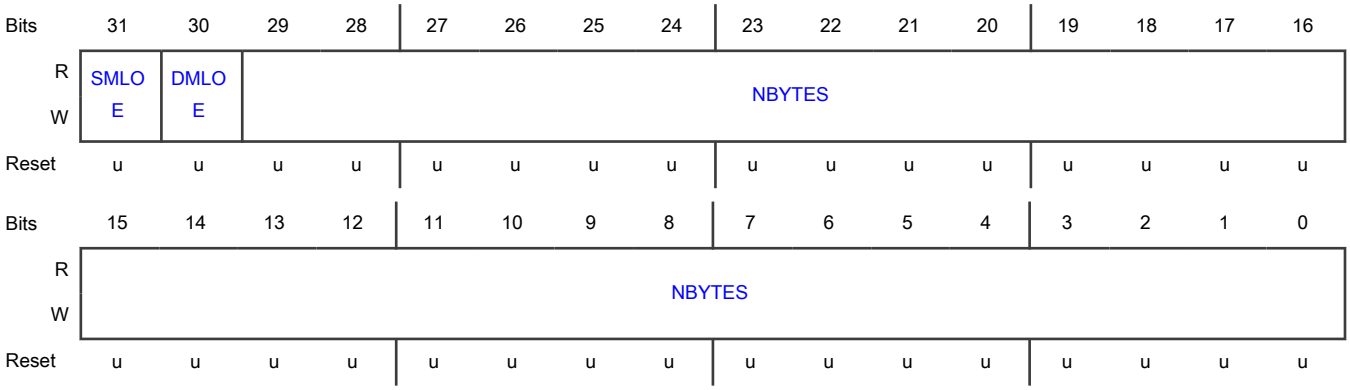
One of two register profiles (this register or TCDn_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether source or destination minor loop mapping is enabled.

TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is defined as follows:

- If SMLOE = 0 and DMLOE = 0, then see the TCDn_NBYTES_MLOFFNO register description.

- If either SMLOE or DMLOE is 1, then see the TCDn_NBYTES_MLOFFYES register description.

Diagram



Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0b - Minor loop offset not applied to SADDR 1b - Minor loop offset applied to SADDR
30 DMLOE	Destination Minor Loop Offset Enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0b - Minor loop offset not applied to DADDR 1b - Minor loop offset applied to DADDR
29-0 NBYTES	Number of Bytes To Transfer Per Service Request Number of bytes to be transferred for each service request of the channel. When a channel activates, the module loads the appropriate TCD contents into the eDMA engine and performs the appropriate reads and writes until the byte transfer count has been reached. This process is normally an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the byte count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major loop iteration count (CITER) is decremented by one and written back to the TCD memory. If the major iteration count is complete, additional processing is performed.

23.6.4.12 TCD Transfer Size with Minor Loop Offsets (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)

Offset

For n = 0 to 15:

Register	Offset
TCDn_NBYTES_MLOFFYES	28h + (n × 1000h)

Function

The TCDn_NBYTES field defines the number of bytes to transfer per service request.

Minor loop offset is an address offset value added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. Minor loop completion occurs when the channel has finished the service request and has transferred NBYTES. Minor loop offsets are enabled by setting either the source enable bit (SMLOE) or the destination enable bit (DMLOE).

The source enable bit (SMLOE) specifies the minor loop offset value (MLOFF) that is to be applied to the source address (TCDn_SADDR) upon minor loop completion. The destination enable bit (DMLOE) specifies the minor loop offset (MLOFF) that is to be applied to the destination address (TCDn_DADDR) upon minor loop completion.

If the major loop is complete, the minor loop offsets are ignored and the major loop address offsets (TCDn_SLAST_SDA and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

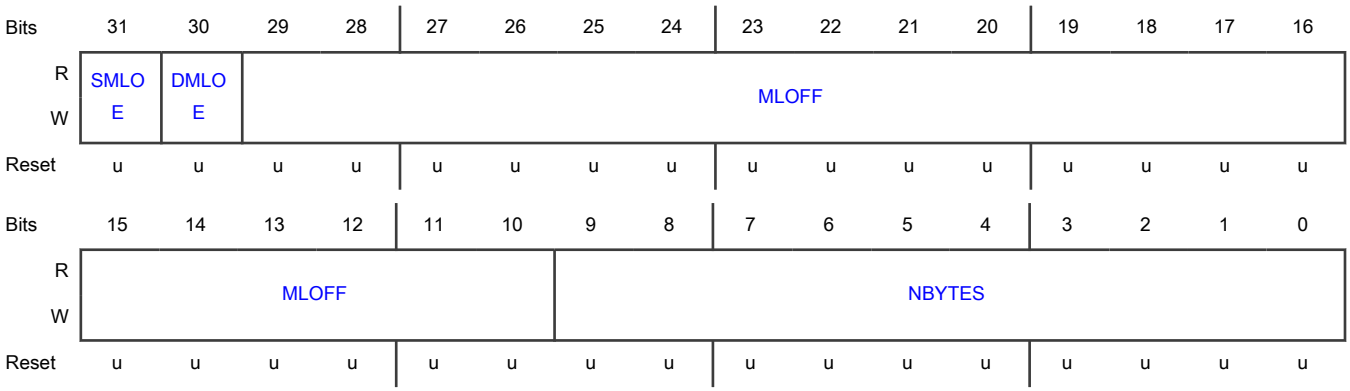
When you enable the minor loop offset overlay (either SMLOE or DMLOE is 1), eDMA redefines TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES. A portion of TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES specifies the sign-extended minor loop offset value (MLOFF). The same offset value (MLOFF) applies to both source and destination minor loop offsets. When the minor loop offset is enabled, you must align it to the transfer size of the source or destination it is associated with. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. If both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

One of two register profiles (this register or TCDn_NBYTES_MLOFFNO) defines the number of bytes to transfer per request. Which register to use depends on whether source or destination minor loop mapping is enabled.

TCDn_NBYTES_MLOFFYES is defined as follows:

- If either minor loop offset is enabled (SMLOE or DMLOE = 1), then see the TCDn_NBYTES_MLOFFYES register description.
- If SMLOE and DMLOE are both 0, then see the TCDn_NBYTES_MLOFFNO register description.

Diagram



Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0b - Minor loop offset not applied to SADDR 1b - Minor loop offset applied to SADDR
30 DMLOE	Destination Minor Loop Offset Enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0b - Minor loop offset not applied to DADDR 1b - Minor loop offset applied to DADDR
29-10 MLOFF	Minor Loop Offset If SMLOE or DMLOE is 1, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
9-0 NBYTES	Number of Bytes To Transfer Per Service Request The number of bytes to be transferred in each service request of the channel. As a channel activates, the module loads the appropriate TCD contents into the eDMA engine and performs the appropriate reads and writes until the minor byte transfer count has been reached. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is complete, additional processing is performed.

23.6.4.13 TCD Last Source Address Adjustment / Store DADDR Address (TCD0_SLAST_SDA - TCD15_SLAST_SDA)

Offset

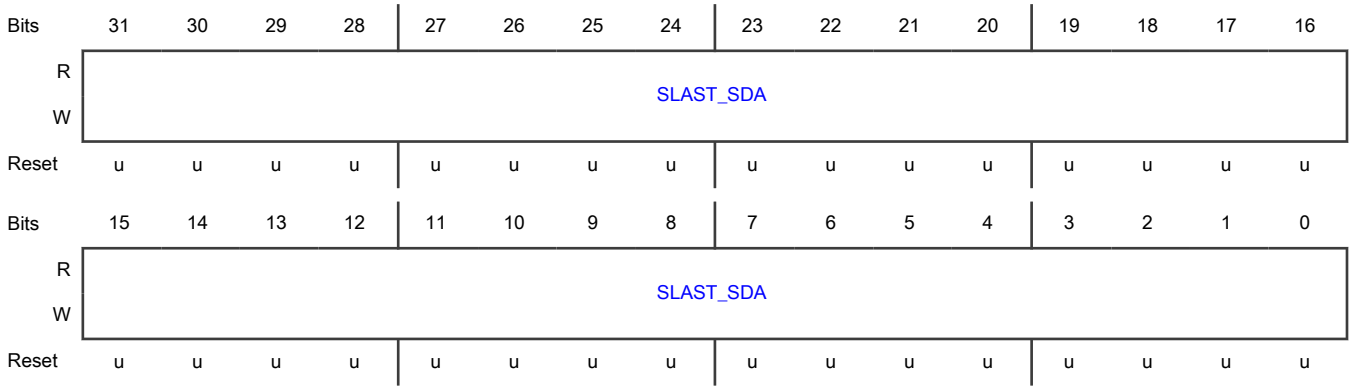
For n = 0 to 15:

Register	Offset
TCDn_SLAST_SDA	2Ch + (n × 1000h)

Function

This register contains the value added to the source address when the major loop is complete. When the store destination address option is enabled, this field provides a pointer to memory for storing the final destination address.

Diagram



Fields

Field	Function
31-0 SLAST_SDA	Last Source Address Adjustment / Store DADDR Address Source last address adjustment or the system memory address for destination address (DADDR) storage. If (TCDn_CSR[ESDA] = 0), then: • Adjustment value is added to the source address at the completion of the major iteration count. This value can be used to restore the source address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final source address adjustment. Otherwise: • This address points to the 32-bit-aligned memory location where the destination address (DADDR) is to be stored in system memory. By saving the final destination address in system memory via the ESDA feature, you are able to compute the size of a variable destination data buffer by simply subtracting the beginning DADDR from the final, saved DADDR. This feature is used together with the scatter/gather operation to prevent the loss of the final DADDR, which is overwritten during the scatter/gather operation. The "Store Destination Address" (SDA) value must be a 32-bit-aligned location because the eDMA forces the lower two address bits of the SLAST_SDA field to zero when ESDA is enabled. The module performs this write operation when the major loop is done; that is, when the major iteration count (CITER) decrements to zero.

23.6.4.14 TCD Destination Address (TCD0_DADDR - TCD15_DADDR)

Offset

For n = 0 to 15:

Register	Offset
TCDn_DADDR	30h + (n × 1000h)

Function

This register contains the address for the write transactions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 DADDR	Destination Address Memory address pointing to the destination data.

23.6.4.15 TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)

Offset

For n = 0 to 15:

Register	Offset
TCDn_DOFF	34h + (n × 1000h)

Function

This register contains the sign-extended value added to Destination Address register after each write transaction.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
15-0 DOFF	Destination Address Signed Offset Sign-extended offset that is applied to the current destination address to form the next-state value as each destination write is completed.

23.6.4.16 TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)

Offset

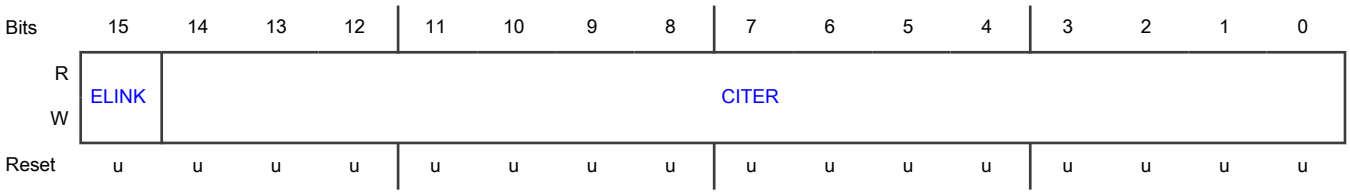
For n = 0 to 15:

Register	Offset
TCDn_CITER_ELINKNO	36h + (n × 1000h)

Function

If TCDn_CITER[ELINK] is 0, the TCDn_CITER register is defined as follows.

Diagram



Fields

Field	Function
15 ELINK	Enable Link As the channel completes the minor loop, this flag enables linking to another channel as defined by the relevant LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel to 1. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of MAJORELINK channel linking. **NOTE** This field must be equal to the BITER[ELINK] field; otherwise, a configuration error is reported. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-0 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the channel finishes a service request and is written back to TCD memory. After the major iteration count is exhausted, the channel performs a number of operations — for example, final source and destination address calculations — and optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.6.4.17 TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)

Offset

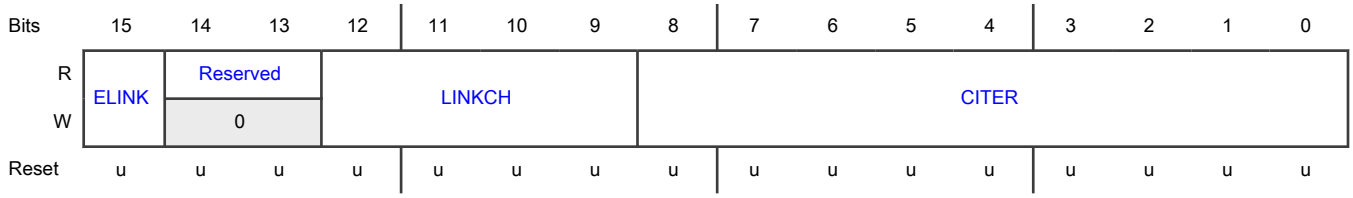
For n = 0 to 15:

Register	Offset
TCDn_CITER_ELINKYES	36h + (n × 1000h)

Function

If TCDn_CITER[ELINK] is 1, the TCDn_CITER register is defined as follows.

Diagram



Fields

Field	Function
15 ELINK	Enable Link As the channel completes the minor loop, this flag enables linking to another channel as defined by the relevant LINKCH field. When enabled, an internal mechanism sets the TCDn_CSR[START] field of the specified channel (LINKCH) upon minor loop completion. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of MAJORELINK channel linking.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field must be equal to the BITER[ELINK] field; otherwise, a configuration error is reported. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-13 —	Reserved
12-9 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted the eDMA engine initiates a channel service request to the channel defined by this field by writing that channel's TCDn_CSR[START] field to 1.
8-0 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the channel finishes a service request and is written back to the TCD memory. After the major iteration count is exhausted, the channel performs a number of operations — for example, final source and destination address calculations — and optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.6.4.18 TCD Last Destination Address Adjustment / Scatter Gather Address (TCD0_DLAST_SGA - TCD15_DLAST_SGA)

Offset

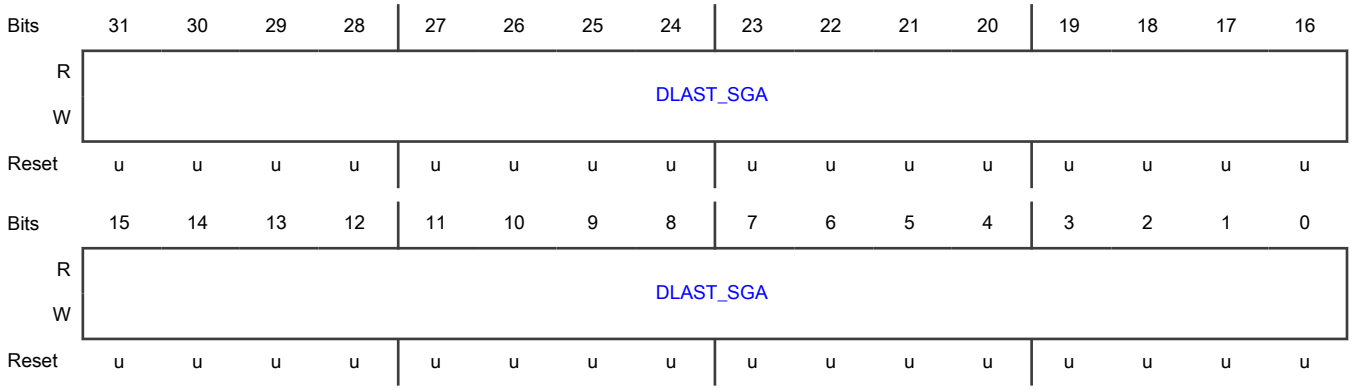
For n = 0 to 15:

Register	Offset
TCDn_DLAST_SGA	38h + (n × 1000h)

Function

This register contains the value added to the destination address when the major loop is complete. When the Scatter/Gather option is enabled, this field provides a pointer to memory for fetching a transfer control descriptor to reprogram the channel.

Diagram



Fields

Field	Function
31-0 DLAST_SGA	Last Destination Address Adjustment / Scatter Gather Address Adjustment of the last destination address or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • Adjustment value is added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo 32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo 32-byte, or else a configuration error is reported.

23.6.4.19 TCD Control and Status (TCD0_CSR - TCD15_CSR)

Offset

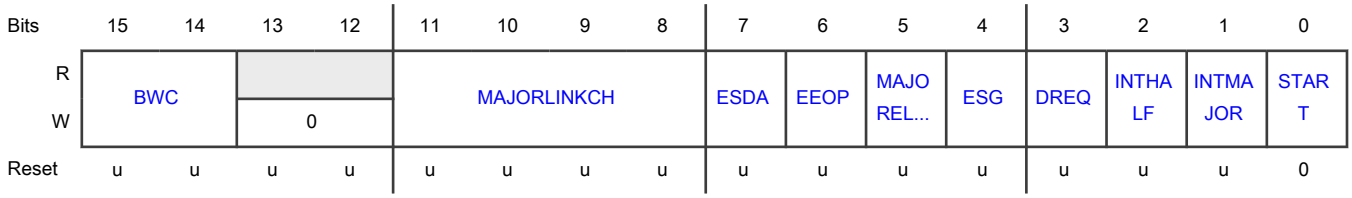
For n = 0 to 15:

Register	Offset
TCDn_CSR	3Ch + (n × 1000h)

Function

This register is used to enable optional features.

Diagram



Fields

Field	Function
15-14 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces eDMA to stall after the completion of each read/write access, to control the bus request bandwidth seen by the system bus interconnect. **NOTE** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00b - No eDMA engine stalls 01b - Reserved 10b - eDMA engine stalls for 4 cycles after each R/W 11b - eDMA engine stalls for 8 cycles after each R/W
13-12 —	Reserved
11-8 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] field to 1.
7 ESDA	Enable Store Destination Address As the channel completes the major loop by either the current iteration counter (CITER) decrementing to 0, or by receiving an enabled end-of-packet signal, this field enables writing the destination address (DADDR) to the address stored in the SLAST_SDA field. The value written to system memory is the last DADDR value prior to the DLAST_SGA offset being applied, or overwritten by an enabled scatter/gather operation. When the ESDA bit is 1, SLAST_SDA contains the write pointer instead of the final source address offset. Because this is a pointer and not a final offset, a last source address offset of zero is applied to SADDR instead of the SLAST_SGA value. 0b - Ability to store destination address to system memory disabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Ability to store destination address to system memory enabled
6 EEOP	Enable End-Of-Packet Processing When enabled by the EEOP field, an end-of-packet hardware input signal directs eDMA to discontinue executing the active channel, and to treat the shutdown as the major-loop-completed event. If the EEOP field is 1, the end-of-packet signal from supported peripherals is accepted. If the EEOP field is 0, the end-of-packet input is ignored. With an end-of-packet retirement, the current TCD destination address (or ESDA-saved destination address), minus the software-saved initial address (DADDR), reflects the total amount of data transferred. 0b - End-of-packet operation disabled 1b - End-of-packet hardware input signal enabled
5 MAJORELINK	Enable Link When Major Loop Complete As the channel completes the major loop, this flag enables linking to another channel defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. **NOTE** To support the dynamic linking coherency model, this field is forced to 0 if written when TCDn_CSR[DONE] is 1. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
4 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses TCDn_DLAST_SGA as a memory pointer to a 0-modulo 32-bit address containing a 32-byte data structure, which is loaded as the transfer control descriptor into local memory. **NOTE** To support the dynamic scatter/gather coherency model, this field is forced to 0 if written when TCDn_CSR[DONE] is 1. 0b - Current channel's TCD is normal format 1b - Current channel's TCD specifies scatter/gather format.
3 DREQ	Disable Request If this flag is 1, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches 0. 0b - No operation. Channel's ERQ field not affected 1b - Clear the ERQ field to 0 upon major loop completion, thus disabling hardware service requests. Channel's ERQ field cleared to 0 when major loop complete
2	Enable Interrupt If Major Counter Half-complete

Table continues on the next page...

Table continued from the previous page...

Field	Function
INTHALF	If this flag is 1, the channel generates an interrupt request by setting the appropriate field in the INT register to 1 when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER = (BITER/2)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes, or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE** If BITER = 1, do not use INTHALF; use INTMAJOR instead. 0b - Halfway point interrupt disabled 1b - Halfway point interrupt enabled
1 INTMAJOR	Enable Interrupt If Major count complete If this flag is 1, the channel generates an interrupt request by setting the appropriate field in the INT register to 1 when the current major iteration count (CITER) reaches 0. 0b - End-of-major loop interrupt disabled 1b - End-of-major loop interrupt enabled
0 START	Channel Start If this flag is 1, the channel is requesting service. The eDMA hardware automatically clears this flag to 0 after the channel begins execution. 0b - Channel not explicitly started 1b - Channel explicitly started via a software-initiated service request

23.6.4.20 TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)

Offset

For n = 0 to 15:

Register	Offset
TCDn_BITER_ELINKNO	3Eh + (n × 1000h)

Function

If the TCDn_BITER[ELINK] field is 0, the TCDn_BITER register is defined as follows.

Diagram

Fields

Field	Function
15 ELINK	Enables Link As the channel completes the minor loop, this flag enables linking to another channel as defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-0 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be set equal to the value in the CITER field. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER must be 0x0001.

23.6.4.21 TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)

Offset

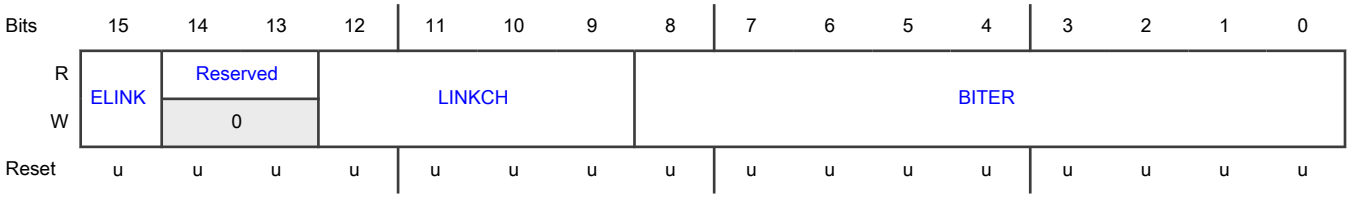
For n = 0 to 15:

Register	Offset
TCDn_BITER_ELINKYES	3Eh + (n × 1000h)

Function

If the TCDn_BITER[ELINK] field is set, the TCDn_BITER register is defined as follows.

Diagram



Fields

Field	Function
15 ELINK	Enable Link As the channel completes the minor loop, this flag enables linking to another channel as defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. 0b - Channel-to-channel linking disabled 1b - Channel-to-channel linking enabled
14-13 —	Reserved
12-9 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] field. **NOTE** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field.
8-0 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be set equal to the value in the CITER field. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER must be 0x0001.

Chapter 24
Wakeup Unit (WUU)

24.1 Chip-specific WUU information

Table 301. Reference links to related information

Topic	Related module	Reference
Full description	WUU	WUU
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

LPO = lp_osc. See Figure 128.

24.1.1 Module instances

This device contains one instance of the WUU module, WUU0.

24.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

24.1.3 WUU sources

The device uses the following inputs as wakeup sources to the WUU module. The WUU_Px inputs are connections to external physical pins. The WUU_MxIF inputs are connections to the internal peripheral interrupt flags from different modules that can operate in low-power modes. The WUU_MxDR inputs are connections to the internal peripheral Asynchronous DMA or module Triggers from different modules that can operate in low-power modes.

NOTE

In addition to the WUU wakeup sources, the device also wakes from low power modes when NMI or RESET pins are enabled and the respective pin is asserted.

Table 302. External pin wakeup sources

WUU input	Chip signal name	Chip pin name
WUU_P0	WUU0_IN0	P0_4
WUU_P1	WUU0_IN1	P0_7
WUU_P2	WUU0_IN2	P0_16
WUU_P3	WUU0_IN3	P0_19
WUU_P4	WUU0_IN4	P0_20
WUU_P5	WUU0_IN5	P0_23
WUU_P6	WUU0_IN6	P1_0

Table continues on the next page...

Table 302. External pin wakeup sources (continued)

WUU input	Chip signal name	Chip pin name
WUU_P7	WUU0_IN7	P1_3
WUU_P8	WUU0_IN8	P1_4
WUU_P9	WUU0_IN9	P1_7
WUU_P10	WUU0_IN10	P1_8
WUU_P11	WUU0_IN11	P1_11
WUU_P12	WUU0_IN12	P1_12
WUU_P13	WUU0_IN13	P1_15
WUU_P14	WUU0_IN14	P1_16
WUU_P15	WUU0_IN15	P1_19
WUU_P16	WUU0_IN16	P2_2
WUU_P17	WUU0_IN17	P2_4
WUU_P18	WUU0_IN18	P4_0
WUU_P19	WUU0_IN19	P4_3
WUU_P20	WUU0_IN20	P4_12
WUU_P21	WUU0_IN21	P4_15
WUU_P22	WUU0_IN22	P3_0
WUU_P23	WUU0_IN23	P3_8
WUU_P24	WUU0_IN24	P3_11
WUU_P25	WUU0_IN25	P3_14
WUU_P26	WUU0_IN26	P3_17
WUU_P27	WUU0_IN27	P3_20
WUU_P28	WUU0_IN28	USB0_DM
WUU_P29	WUU0_IN29	USB0_DP

Table 303. On-chip module sources

WUU input	Module source
WUU_M0IF	SPC
WUU_M1IF	VBAT
WUU_M2IF	RTC
WUU_M3IF	TDET
WUU_M4IF	GPIO5 Interrupt 0
WUU_M5IF	GPIO5 Interrupt 1
WUU_M6IF	LPTMR0

Table continues on the next page...

Table 303. On-chip module sources (continued)

WUU input	Module source
WUU_M7IF	LPTMR1
WUU_M8IF	CMP0
WUU_M9IF	CMP1
WUU_M0DR	GPIO5 DMA 0
WUU_M1DR	GPIO5 DMA 1
WUU_M2DR	GPIO5 Trigger0
WUU_M3DR	GPIO5 Trigger1
WUU_M4DR	LPTMR0 DMA
WUU_M5DR	LPTMR1 DMA
WUU_M6DR	LPTMR0 Trigger
WUU_M7DR	LPTMR1 Trigger
WUU_M8DR	CMP0 DMA
WUU_M9DR	CMP1 DMA

24.2 Overview

WUU facilitates in selecting external pins and on-chip modules as interrupt wake-up sources from Power Down/Deep Power Down power modes. You can also configure the external pins to act as interrupt wake-up sources in all power modes. WUU helps pins and modules generate a temporary wake-up from Power Down mode for servicing DMA or trigger events. Digital filtering is also available for the external wake-up pins.

See the chip-specific WUU information section for input sources to WUU.

24.2.1 Block diagram

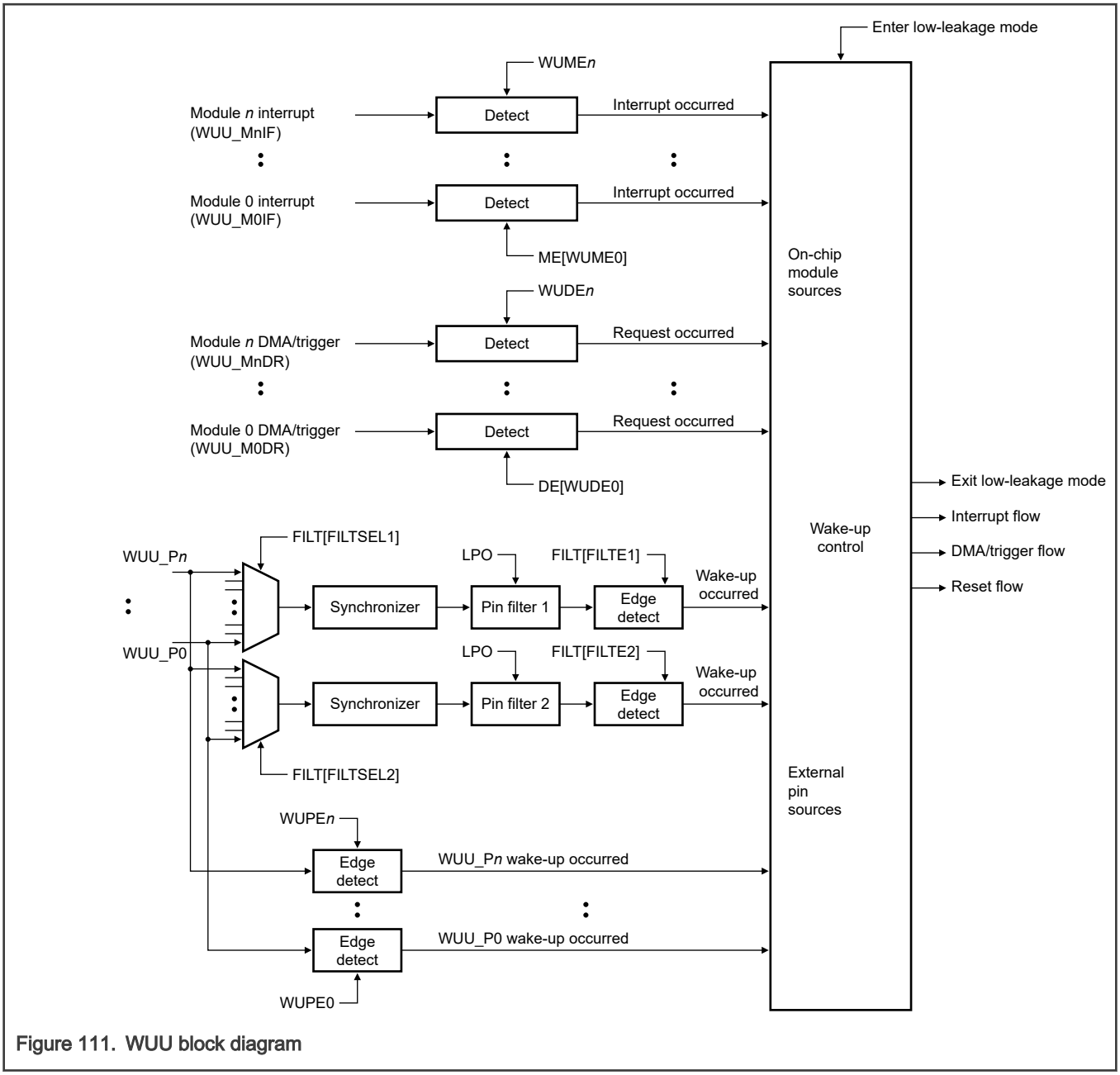


Figure 111. WUU block diagram

24.2.2 Features

- Supports external input pins and on-chip modules with individual enable bits to wake the chip from low-leakage modes.
- Contains input sources that may be external pins or from on-chip modules capable of running in Power Down or Deep Power Down power modes. See the chip-specific WUU information for the wake-up input sources for this chip.
- Supports configuring on-chip modules as DMA or trigger sources for temporary wake-up from Power Down mode.
- Supports configuring external pins as DMA request or trigger sources for temporary wake-up from Power Down mode.
- Provides external input pins programmable for falling-edge, rising-edge, or any edge detection.
- Provides optional digital filters to qualify an external pin detect. Disabling the LPO clock bypasses and disables filters.

- Filters all external input pins; but only 2 filters are available at a given time.
- Enables external input pin and filter detection in all power modes (not limited to Power Down/Deep Power Down power modes).

24.3 Functional description

24.3.1 Operation

WUU allows on-chip modules and external input pins as a source of wake-up from low-leakage modes.

WUU contains pin enables for each external pin and on-chip module. For each external pin, you can disable or select the edge type for the wake-up with the following options:

- Falling edge
- Rising edge
- Either edge

Exit due to a pin event triggers a WUU interrupt after wake-up is complete. When enabling an external pin as a wake-up source, configure the pin as an input pin. You can also configure a pin to generate an interrupt from a Power Down/Deep Power Down mode or a DMA/trigger request for temporary wake-up from Power Down mode. Detection logic for a pin can optionally remain enabled during all power modes using Pin Mode Configuration (PMC).

On the basis of the LPO clock, WUU implements optional 3-cycle glitch filters. Detected external pin must remain asserted until the enabled glitch filter times out. Additional latency of up to two cycles is due to synchronization, which results in a total of up to five cycles of delay before the detect circuit alerts the system to the wake-up event when the filter function is enabled. Two wake-up detect filters are available for selected external pins. Glitch filtering is not provided for the on-chip modules. You can configure a filter to generate an interrupt from Power Down/Deep Power Down mode or a DMA/trigger request for temporary wake-up from Power Down mode. Detection logic for a filter can optionally remain enabled during all power modes using Pin Filter Mode Configuration (FMC).

For on-chip module interrupts, ME[WUME*n*] enables the associated module interrupt as a wake-up source. Exit due to a module interrupt will not trigger a WUU interrupt.

For on-chip module DMA/trigger requests, DE[WUDE*n*] enables the associated module DMA/trigger request as a wake-up source. Exit due to a module DMA/trigger request does not trigger a WUU interrupt.

24.3.2 Modes of operation

WUU becomes functional on entry into a low-leakage power mode. After recovery from Power Down mode, WUU is immediately disabled. After recovery from Deep Power Down mode, WUU continues to detect wake-up events until you have reinitialized the system and deasserted isolation. Detection of external pin/filter events can also optionally remain enabled in all power modes using Pin Mode Configuration (PMC)/Pin Filter Mode Configuration (FMC).

24.3.2.1 Power Down mode

Wake-up events due to either an external pin input (WUU_Px) or an on-chip module interrupt (WUU_MxIF) result in a CPU interrupt flow to begin user code execution. The source of the wake-up determines the subsequent path of code execution:

- Pin events trigger the WUU interrupt service routine.
- Module interrupts trigger that same module's interrupt service routine.

NOTE

The interrupt controller must not mask the WUU interrupt to avoid a scenario where the system does not fully recover from Power Down mode after an external pin event.

Wake-up events due to an on-chip module request (WUU_MxDR) DMA/trigger request result in a temporary exit from Power Down mode to allow the request to be serviced when the CPU clock remains gated. The system reenters Power Down mode after servicing the request.

Wake-up events due to an external pin (WUU_Px) DMA/trigger request result in a temporary exit from Power Down mode to allow the request to be serviced when the CPU clock remains gated. The system reenters Power Down mode after servicing the request.

24.3.2.2 Non–low-leakage modes

WUU is inactive in all non–low-leakage modes, with the exception of external pin/filter detection which can explicitly be kept enabled during all power modes using Pin Mode Configuration (PMC)/Pin Filter Mode Configuration (FMC). WUU chips are accessible in non–low-leakage modes and are available for configuring and reading status when bus transactions are possible.

24.3.3 Clocking

During chip low-power operation, in order to provide the wake-up function for the chip, ensure that clocking is available for WUU by the LPO clock. See the chip-specific WUU information for the chip clock connection details.

24.3.4 Reset

WUU resets after VSYS Warm Reset. This is the global chip reset, but it does not include the reset that asserts due to wake-up from low-power mode.

24.3.5 Interrupts

You can enable on-chip module interrupts as wake-up sources in Module Interrupt Enable (ME).

24.3.6 DMA

You can enable on-chip DMA requests as wake-up sources in Module DMA/Trigger Enable (DE).

With external pin inputs, you can generate a DMA request using either rising edge, falling edge, or both edge detection in Pin DMA/Trigger Configuration 1 (PDC1) and Pin DMA/Trigger Configuration 2 (PDC2).

24.4 External signals

Table 304. External signals

Signal	Description	I/O
WUU_Pn	External pin wake-up inputs; can be enabled to detect a rising edge, a falling edge, or any change.	I
WUU_MnIF	On-chip module interrupt flags	I
WUU_MnDR	On-chip module DMA/trigger requests	I

24.5 Initialization

For an enabled module wake-up input, clear the module's flag before entering Power Down or Deep Power Down mode to avoid an immediate exit from the mode.

Clear the flags associated with external input pins, filtered and unfiltered, prior to entry to Power Down or Deep Power Down mode.

When an external wake-up pin filter is first enabled in Pin Filter (FILT), filter operation begins immediately. After enabling an external pin filter or changing its source pin, wait for at least five LPO clock cycles to allow the filter to initialize before entering Power Down or Deep Power Down mode or before enabling filter detection for all power modes in Pin Filter Mode Configuration (FMC).

NOTE

After recovering from a Deep Power Down mode, you must restore the chip configuration before relaxing any power domain isolation controls. In particular, to avoid any WUU flag from being falsely set, restore the pin configuration for the enabled WUU wake-up pins before removing any voltage isolation between power domains.

24.6 Application information

To enable an external pin as a wake-up source in low-power modes, follow these steps:

1. If the flag is not 0, write 1 to PF[WUF*n*].
2. Configure the PDC*m*[WUPDC*n*] as either an interrupt or DMA request.
3. Configure the PE*m*[WUPE*n*] for rising edge, falling edge, or both edge detection.

When you have configured an external pin to be an active wake-up source in all power modes (PMC[WUPMC*n*] = 1), you must not change the corresponding wake-up pin enable (PE*m*[WUPE*n*]) and wake-up pin configuration (PDC*m*[WUPDC*n*]) settings for that pin. To modify the PE*m*[WUPE*n*] or PDC*m*[WUPDC*n*] settings, follow these steps:

1. Write 0 to PMC[WUPMC*n*] for that pin.
2. Update the corresponding PE*m*[WUPE*n*] or PDC*m*[WUPDC*n*] settings as needed.
3. Write 1 to PMC[WUPMC*n*] for that pin.

24.7 Memory map and register definition

This section includes the WUU module memory map and detailed descriptions of all registers.

24.7.1 WUU register descriptions

NOTE

You can write to WUU registers only in Supervisor mode. A bus error results from write accesses in User mode.

VSYS Warm Reset resets all WUU registers.

24.7.1.1 WUU memory map

WUU0 base address: 4004_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0100_0001h
4h	Parameter (PARAM)	32	R	2020_2002h
8h	Pin Enable 1 (PE1)	32	RW	0000_0000h
Ch	Pin Enable 2 (PE2)	32	RW	0000_0000h
18h	Module Interrupt Enable (ME)	32	RW	0000_0000h
1Ch	Module DMA/Trigger Enable (DE)	32	RW	0000_0000h
20h	Pin Flag (PF)	32	RW	0000_0000h
30h	Pin Filter (FILT)	32	RW	0000_0000h
38h	Pin DMA/Trigger Configuration 1 (PDC1)	32	RW	0000_0000h
3Ch	Pin DMA/Trigger Configuration 2 (PDC2)	32	RW	0000_0000h
48h	Pin Filter DMA/Trigger Configuration (FDC)	32	RW	0000_0000h
50h	Pin Mode Configuration (PMC)	32	RW	0000_0000h
58h	Pin Filter Mode Configuration (FMC)	32	RW	0000_0000h

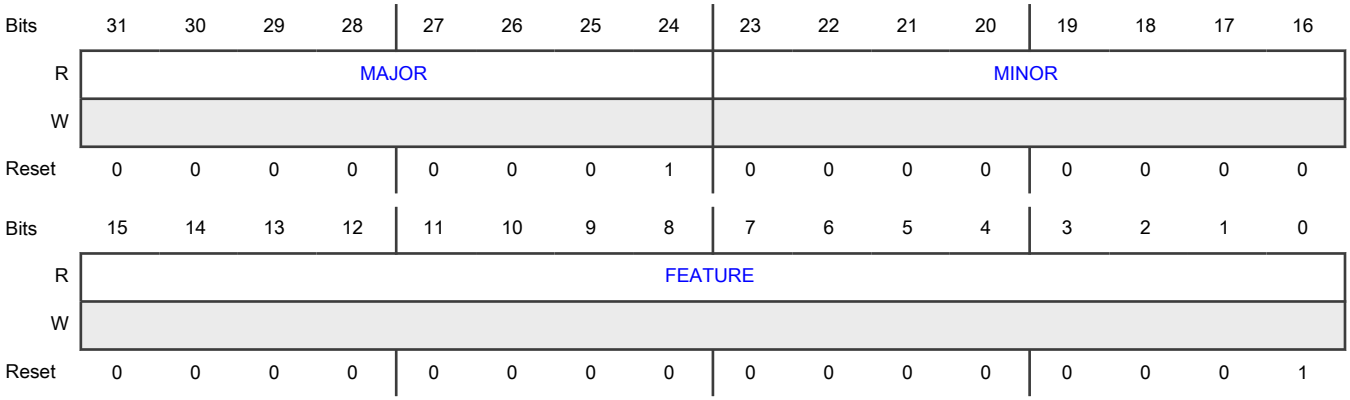
24.7.1.2 Version ID (VERID)

Offset

Register	Offset
VERID	0h

Function
Contains version numbers for the module design and feature set.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Specifies the major version number for the module specification.
23-16 MINOR	Minor Version Number Specifies the minor version number for the module specification.
15-0 FEATURE	Feature Specification Number Specifies the feature set number. 0000_0000_0000_0000b - Standard features implemented 0000_0000_0000_0001b - Support for DMA/Trigger generation from wake-up pins and filters enabled. Support for external pin/filter detection during all power modes enabled. All other values are reserved.

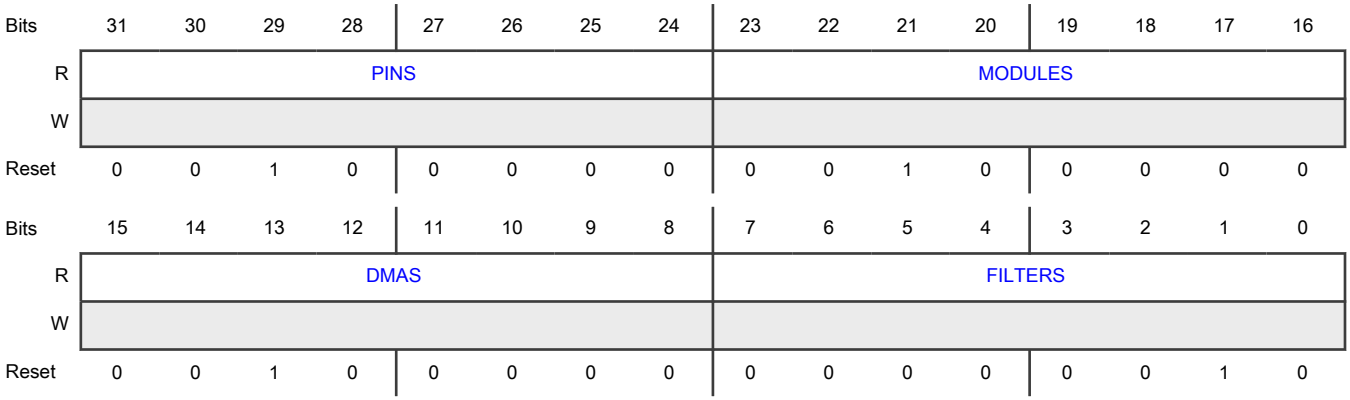
24.7.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function
Contains parameter values implemented in the module.

Diagram



Fields

Field	Function
31-24 PINS	Pin Number Indicates the number of pin wake-up sources supported.
23-16 MODULES	Module Number Indicates the number of module wake-up sources.
15-8 DMAS	DMA Number Indicates the number of DMA wake-up sources.
7-0 FILTERS	Filter Number Indicates the number of pin filters.

24.7.1.4 Pin Enable 1 (PE1)

Offset

Register	Offset
PE1	8h

Function

Enables and selects the edge detect type for the available external wake-up input pins in the range from WUU_P0 to WUU_P15.

NOTE

- Do not modify the value of PE1[WUPE*n*] when its corresponding PMC[WUPMC*n*] = 1.
- VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WUPE15		WUPE14		WUPE13		WUPE12		WUPE11		WUPE10		WUPE9		WUPE8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WUPE7		WUPE6		WUPE5		WUPE4		WUPE3		WUPE2		WUPE1		WUPE0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 WUPE15	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
29-28 WUPE14	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
27-26 WUPE13	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
25-24 WUPE12	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
23-22 WUPE11	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
21-20 WUPE10	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
19-18 WUPE9	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
17-16 WUPE8	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
15-14 WUPE7	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
13-12 WUPE6	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
11-10 WUPE5	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Enable (detect on any edge)
9-8 WUPE4	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
7-6 WUPE3	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
5-4 WUPE2	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 WUPE1	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
1-0 WUPE0	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)

24.7.1.5 Pin Enable 2 (PE2)

Offset

Register	Offset
PE2	Ch

Function

Contains the field to enable and select the edge detect type for the available external wake-up input pins in the range from WUU_P16 to WUU_P31.

NOTE

- Do not modify the value of PE2[WUPE*n*] when its corresponding PMC[WUPMC*n*] = 1.
- VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						WUPE28		WUPE27		WUPE26		WUPE25		WUPE24	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WUPE23		WUPE22		WUPE21		WUPE20		WUPE19		WUPE18		WUPE17		WUPE16	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25-24 WUPE28	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
23-22 WUPE27	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)

Table continues on the next page...

Table continued from the previous page...

Field	Function
21-20 WUPE26	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
19-18 WUPE25	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
17-16 WUPE24	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
15-14	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection.

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUPE23	• For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
13-12 WUPE22	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
11-10 WUPE21	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
9-8 WUPE20	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
7-6 WUPE19	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
5-4 WUPE18	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
3-2 WUPE17	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)
1-0 WUPE16	Wake-up Pin Enable for WUU_Pn Enables the external input pin for wake-up and configures the edge detection. • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level. • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level. • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge. 00b - Disable 01b - Enable (detect on rising edge or high level) 10b - Enable (detect on falling edge or low level) 11b - Enable (detect on any edge)

24.7.1.6 Module Interrupt Enable (ME)

Offset

Register	Offset
ME	18h

Function

Contains the bits to enable an on-chip module interrupt as a wake-up source.

NOTE

VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	WUME9	WUME8	WUME7	WUME6	WUME5	WUME4	WUME3	WUME2	WUME1	WUME0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 —	Reserved
24 —	Reserved
23 —	Reserved
22 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 —	Reserved
20 —	Reserved
19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15 —	Reserved
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10 —	Reserved
9 WUME9	Module Interrupt Wake-up Enable for Module 9 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
8	Module Interrupt Wake-up Enable for Module 8

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUME8	Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
7 WUME7	Module Interrupt Wake-up Enable for Module 7 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
6 WUME6	Module Interrupt Wake-up Enable for Module 6 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
5 WUME5	Module Interrupt Wake-up Enable for Module 5 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
4 WUME4	Module Interrupt Wake-up Enable for Module 4 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
3 WUME3	Module Interrupt Wake-up Enable for Module 3 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
2 WUME2	Module Interrupt Wake-up Enable for Module 2 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable
1 WUME1	Module Interrupt Wake-up Enable for Module 1 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 WUME0	Module Interrupt Wake-up Enable for Module 0 Enables an on-chip module interrupt as a wake-up source input. 0b - Disable 1b - Enable

24.7.1.7 Module DMA/Trigger Enable (DE)

Offset

Register	Offset
DE	1Ch

Function

Contains the bits to enable an on-chip module DMA/trigger request as a wake-up source.

NOTE

VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	WUDE9	WUDE8	WUDE7	WUDE6	WUDE5	WUDE4	WUDE3	WUDE2	WUDE1	WUDE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 —	Reserved
24 —	Reserved
23 —	Reserved
22 —	Reserved
21 —	Reserved
20 —	Reserved
19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10 —	Reserved
9 WUDE9	DMA/Trigger Wake-up Enable for Module 9 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
8 WUDE8	DMA/Trigger Wake-up Enable for Module 8 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
7 WUDE7	DMA/Trigger Wake-up Enable for Module 7 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
6 WUDE6	DMA/Trigger Wake-up Enable for Module 6 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
5 WUDE5	DMA/Trigger Wake-up Enable for Module 5 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
4 WUDE4	DMA/Trigger Wake-up Enable for Module 4 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
3 WUDE3	DMA/Trigger Wake-up Enable for Module 3 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
2 WUDE2	DMA/Trigger Wake-up Enable for Module 2 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
1 WUDE1	DMA/Trigger Wake-up Enable for Module 1 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable
0 WUDE0	DMA/Trigger Wake-up Enable for Module 0 Enables an on-chip module DMA/trigger request as a wake-up source. 0b - Disable 1b - Enable

24.7.1.8 Pin Flag (PF)

Offset

Register	Offset
PF	20h

Function

Contains the wake-up flags indicating which wake-up source caused the chip to exit Power Down (if the corresponding PMC[WUPMC*n*] = 1) or Deep Power Down mode. For Power Down mode, this is the source causing the CPU interrupt flow. For Deep Power Down mode, this is the source causing the chip reset flow.

To clear a flag, write a 1 to the corresponding PF[WUF*n*]. If set, the wake-up flag (WUF*n*) remains set even if the associated pin wake-up is disabled in its PE*m*[WUPE*n*] field.

NOTE

VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			WUF28	WUF27	WUF26	WUF25	WUF24	WUF23	WUF22	WUF21	WUF20	WUF19	WUF18	WUF17	WUF16
W				W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WUF15	WUF14	WUF13	WUF12	WUF11	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 WUF28	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
27 WUF27	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
26 WUF26	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
25 WUF25	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes

Table continues on the next page...

Table continued from the previous page...

Field	Function
24 WUF24	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
23 WUF23	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
22 WUF22	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
21 WUF21	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
20 WUF20	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
19 WUF19	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
18 WUF18	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
17 WUF17	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Yes
16 WUF16	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
15 WUF15	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
14 WUF14	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
13 WUF13	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
12 WUF12	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
11 WUF11	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
10 WUF10	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
9 WUF9	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No 1b - Yes
8 WUF8	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
7 WUF7	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
6 WUF6	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
5 WUF5	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
4 WUF4	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
3 WUF3	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
2 WUF2	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
1	Wake-up Flag for WUU_Pn

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUF1	Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
0 WUF0	Wake-up Flag for WUU_Pn Indicates that an enabled external wake-up pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes

24.7.1.9 Pin Filter (FILT)

Offset

Register	Offset
FILT	30h

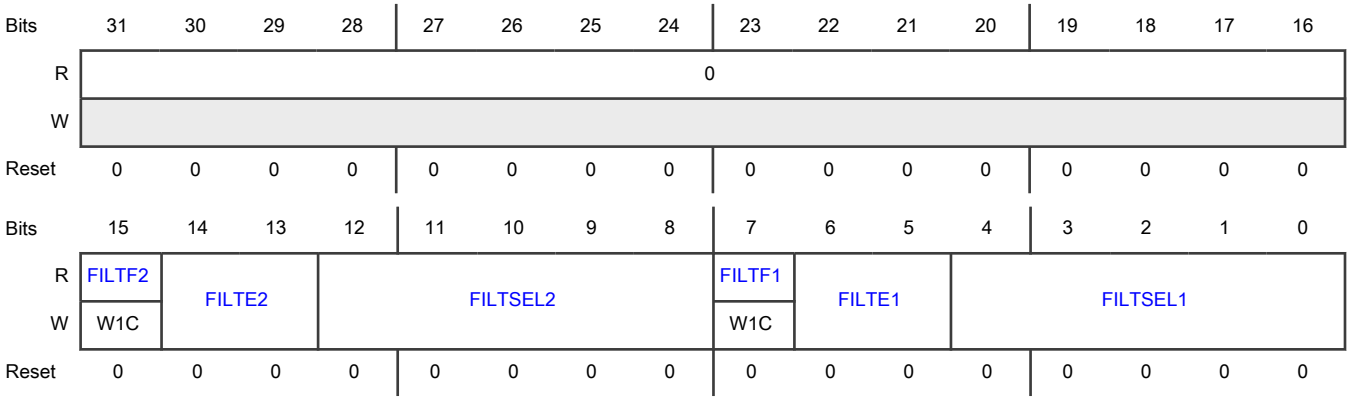
Function

Contains control and status fields to enable and configure the digital filter features for an external wake-up pin.

NOTE

VSYS Warm Reset resets this register.

Diagram



Fields

Field	Function
31-16	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
15 FILTF2	Filter 2 Flag Indicates that the filtered external pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
14-13 FILTE2	Filter 2 Enable Enables the filter for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge 00b - Disable 01b - Enable (Detect on rising edge or high level) 10b - Enable (Detect on falling edge or low level) 11b - Enable (Detect on any edge)
12-8 FILTSEL2	Filter 2 Pin Select Selects and filters the external pin WUU_Pn, where *n* equals the value programmed into FILTSEL2.
7 FILTF1	Filter 1 Flag Indicates that the filtered external pin was a source of exiting a low-leakage power mode. 0b - No 1b - Yes
6-5 FILTE1	Filter 1 Enable Enables the filter for wake-up. This field configures the edge detection as follows: • For field value = 01b, when configured as an interrupt/DMA request: Detect on rising edge. When configured as a trigger request: Detect on high level • For field value = 10b, when configured as an interrupt/DMA request: Detect on falling edge. When configured as a trigger request: Detect on low level • For field value = 11b, when configured as an interrupt/DMA request: Detect on any edge 00b - Disable 01b - Enable (Detect on rising edge or high level) 10b - Enable (Detect on falling edge or low level) 11b - Enable (Detect on any edge)

Table continues on the next page...

Table continued from the previous page...

Field	Function
4-0 FILTSEL1	Filter 1 Pin Select Selects and filters the external pin WUU_Pn, where *n* equals the value programmed into FILTSEL1.

24.7.1.10 Pin DMA/Trigger Configuration 1 (PDC1)

Offset

Register	Offset
PDC1	38h

Function

Configures the available external wake-up input pins in the range from WUU_P0 to WUU_P15 to generate an interrupt, DMA, or a trigger request when detected.

- When configured as an interrupt, the interrupt is asserted when the edge programmed in Pin Enable 1 (PE1) is detected and remains asserted until the corresponding flag is cleared in Pin Flag (PF).
- When configured as a DMA request, the request is asserted when the edge programmed in Pin Enable 1 (PE1) is detected and remains asserted until either the corresponding flag is cleared in Pin Flag (PF) or until the requested DMA transfer is complete.
- When configured as a trigger request, the trigger is asserted when the level programmed in Pin Enable 1 (PE1) is detected and remains asserted when the input pin is asserted. The corresponding flag in Pin Flag (PF) will not be set.

NOTE

- Do not modify the value of PDC*m*[WUPDC*n*] field when its corresponding PMC[WUPMC*n*] = 1.
- VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WUPDC15		WUPDC14		WUPDC13		WUPDC12		WUPDC11		WUPDC10		WUPDC9		WUPDC8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WUPDC7		WUPDC6		WUPDC5		WUPDC4		WUPDC3		WUPDC2		WUPDC1		WUPDC0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 WUPDC15	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
29-28 WUPDC14	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
27-26 WUPDC13	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
25-24 WUPDC12	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
23-22 WUPDC11	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
21-20 WUPDC10	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
19-18 WUPDC9	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
17-16 WUPDC8	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
15-14 WUPDC7	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
13-12 WUPDC6	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
11-10 WUPDC5	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - DMA request 10b - Trigger event 11b - Reserved
9-8 WUPDC4	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
7-6 WUPDC3	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
5-4 WUPDC2	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
3-2 WUPDC1	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
1-0 WUPDC0	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Trigger event 11b - Reserved

24.7.1.11 Pin DMA/Trigger Configuration 2 (PDC2)

Offset

Register	Offset
PDC2	3Ch

Function

Configures the available external wake-up input pins in the range from WUU_P16 to WUU_P31 to generate an interrupt, DMA, or a trigger request when detected.

- When configured as an interrupt, the interrupt is asserted when the edge programmed in Pin Enable 2 (PE2) is detected and remains asserted until the corresponding flag is cleared in Pin Flag (PF).
- When configured as a DMA request, the request is asserted when the edge programmed in Pin Enable 2 (PE2) is detected and remains asserted until either the corresponding flag is cleared in Pin Flag (PF) or until the requested DMA transfer is complete.
- When configured as a trigger request, the trigger is asserted when the level programmed in Pin Enable 2 (PE2) is detected and remains asserted when the input pin is asserted. The corresponding flag in Pin Flag (PF) will not be set.

NOTE

- Do not modify the value of PDC*m*[WUPDC*n*] field when its corresponding PMC[WUPMC*n*] = 1.
- VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WUPDC31		WUPDC30		WUPDC29		WUPDC28		WUPDC27		WUPDC26		WUPDC25		WUPDC24	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WUPDC23		WUPDC22		WUPDC21		WUPDC20		WUPDC19		WUPDC18		WUPDC17		WUPDC16	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 WUPDC31	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
29-28 WUPDC30	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
27-26 WUPDC29	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
25-24 WUPDC28	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
23-22 WUPDC27	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
21-20 WUPDC26	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
19-18 WUPDC25	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
17-16 WUPDC24	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
15-14 WUPDC23	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
13-12 WUPDC22	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
11-10 WUPDC21	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - DMA request 10b - Trigger event 11b - Reserved
9-8 WUPDC20	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
7-6 WUPDC19	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
5-4 WUPDC18	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
3-2 WUPDC17	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved
1-0 WUPDC16	Wake-up Pin Configuration for WUU_Pn Configures an external pin as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Trigger event 11b - Reserved

24.7.1.12 Pin Filter DMA/Trigger Configuration (FDC)

Offset

Register	Offset
FDC	48h

Function

Configures the external pin filters to generate an interrupt, DMA, or a trigger request when detected.

- When configured as an interrupt, the interrupt is asserted when the edge programmed in FILT*m*[FILTE*n*] is detected and remains asserted until the corresponding FILTF*n* flag is cleared.
- When configured as a DMA request, the request is asserted when the edge programmed in FILT*m*[FILTE*n*] is detected and remains asserted until either the corresponding FILTF*n* flag is cleared or until the requested DMA transfer is complete.
- When configured as a trigger request, the trigger is asserted when the level programmed in Pin Filter (FILT) is detected and remains asserted when the input pin is asserted. The corresponding FILTF*n* flag will not be set.

NOTE

VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												FILTC2		FILTC1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3-2: FILTC2 1-0: FILTC1	Filter Configuration for FILTn Configures a filter as an interrupt, DMA, or trigger wake-up source. 00b - Interrupt 01b - DMA request 10b - Trigger event 11b - Reserved

24.7.1.13 Pin Mode Configuration (PMC)

Offset

Register	Offset
PMC	50h

Function

Configures the detection logic for the available external wake-up input pins to remain enabled during all power modes, not just during Power Down/ mode.

NOTE

VSYS Warm Reset resets this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WUPMC31	WUPMC30	WUPMC29	WUPMC28	WUPMC27	WUPMC26	WUPMC25	WUPMC24	WUPMC23	WUPMC22	WUPMC21	WUPMC20	WUPMC19	WUPMC18	WUPMC17	WUPMC16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WUPMC15	WUPMC14	WUPMC13	WUPMC12	WUPMC11	WUPMC10	WUPMC9	WUPMC8	WUPMC7	WUPMC6	WUPMC5	WUPMC4	WUPMC3	WUPMC2	WUPMC1	WUPMC0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Wake-up Pin Mode Configuration for WUU_Pn

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUPMC31	Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
30 WUPMC30	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
29 WUPMC29	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
28 WUPMC28	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
27 WUPMC27	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
26 WUPMC26	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
25	Wake-up Pin Mode Configuration for WUU_Pn

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUPMC25	Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
24 WUPMC24	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
23 WUPMC23	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
22 WUPMC22	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
21 WUPMC21	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
20 WUPMC20	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
19	Wake-up Pin Mode Configuration for WUU_Pn

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUPMC19	Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
18 WUPMC18	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
17 WUPMC17	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
16 WUPMC16	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
15 WUPMC15	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
14 WUPMC14	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
13	Wake-up Pin Mode Configuration for WUU_Pn

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUPMC13	Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
12 WUPMC12	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
11 WUPMC11	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
10 WUPMC10	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
9 WUPMC9	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
8 WUPMC8	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
7	Wake-up Pin Mode Configuration for WUU_Pn

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUPMC7	Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
6 WUPMC6	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
5 WUPMC5	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
4 WUPMC4	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
3 WUPMC3	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
2 WUPMC2	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
1	Wake-up Pin Mode Configuration for WUU_Pn

Table continues on the next page...

Table continued from the previous page...

Field	Function
WUPMC1	Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).
0 WUPMC0	Wake-up Pin Mode Configuration for WUU_Pn Configures an external wake-up pin to provide active detection during all power modes. 0b - Active only during a low-leakage mode. You can modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn). 1b - Active during all power modes. Do not modify the corresponding fields within Pin Enable (PEn) or Pin DMA/Trigger Configuration (PDCn).

24.7.1.14 Pin Filter Mode Configuration (FMC)

Offset

Register	Offset
FMC	58h

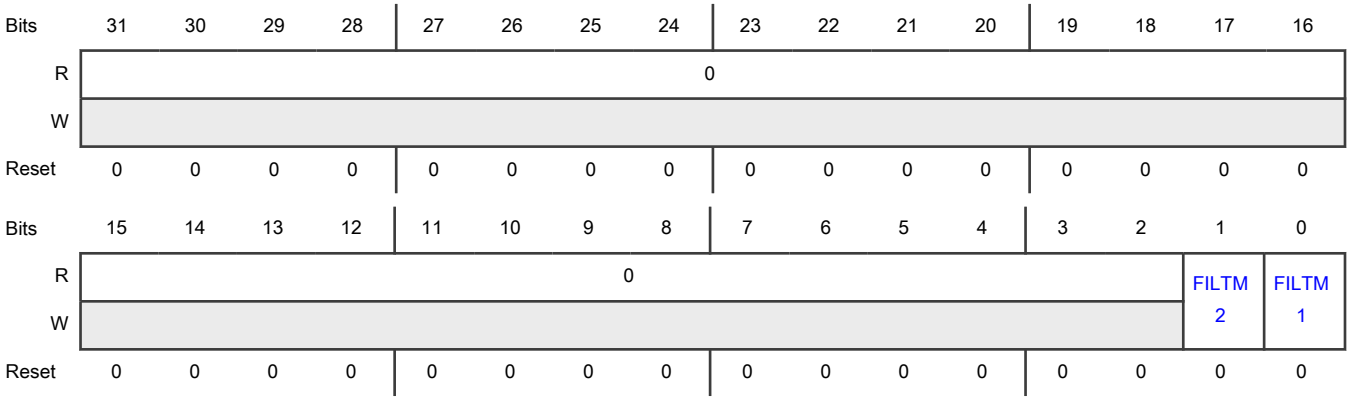
Function

Configures the detection logic for external pin filters to remain enabled during all power modes, not just during Power Down/Deep Power Down mode.

NOTE

VSYS Warm Reset resets this register.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 FILTMn	Filter Mode for FILTn Configures an external wake-up pin filter to provide active detection during all power modes. 0b - Active only during Power Down/Deep Power Down mode 1b - Active during all power modes

Chapter 25
Event Generator (EVTG)

25.1 Chip-specific EVTG information

Table 305. Reference links to related information

Topic	Related module	Reference
Full description	EVTG	EVTG
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See EVTG_TRIGn registers in INPUTMUX

25.1.1 Module instances

This device contains one instance of the EVTG module, EVTG0.

25.2 Overview

EVTG includes two parts:

- Two AND/OR/INVERT (known as the AOI) modules
- One configurable flip-flop

It supports the generation of a configurable number of Event signals. The two AOI combinational expressions share the four associated EVTG inputs: An, Bn, Cn, and Dn. You can configure the flip-flop to make the two expressions act as the Reset port, Set port or D port, CLK port or go through to EVTG output with the flip-flop bypassed.

This module is a slave peripheral module-connecting event input indicators from various chip modules and generates event output signals that you can route to an inter-peripheral crossbar switch or other peripherals. You can access its programming model through the standard IPS (sky blue) slave interface. EVTG is configurable in the integrated AOI functionality and flip-flop variety.

25.2.1 Block diagram

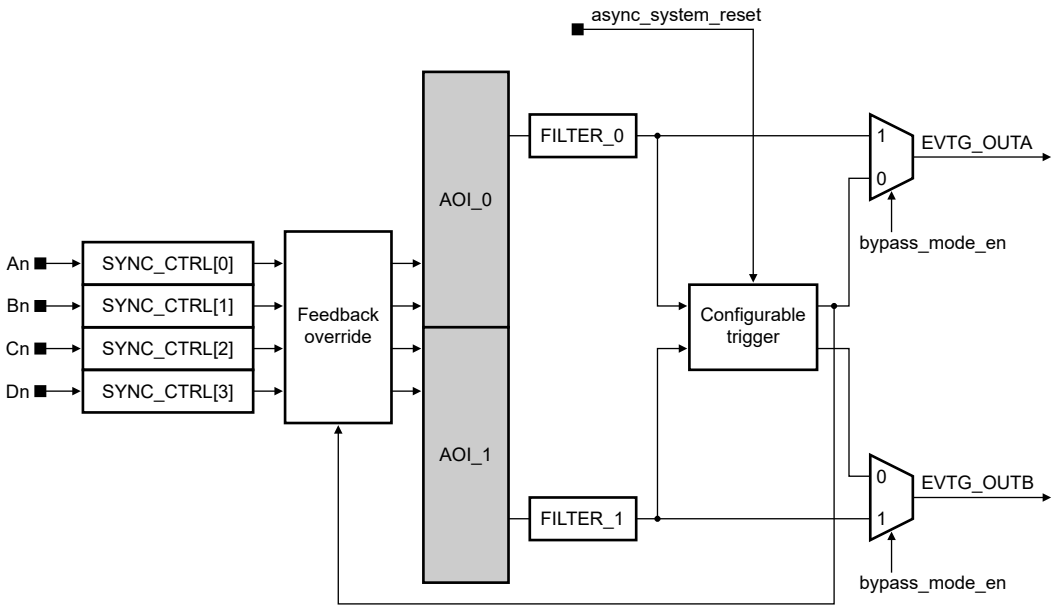


Figure 112. Block diagram

25.2.2 Features

- Includes a highly programmable module for creating combinational boolean events
  - Each EVTG consists of four inputs and two outputs.
  - Each AOI evaluates a combinational boolean expression as the sum of four products, where each product term includes all four selected input sources available as true or complement values.
  - Each EVTG consists of two groups of AOI to generate two combinational expressions.
  - The two outputs can operate as hardware trigger signals or for other purposes.
- Configures one flexible FF as RS, D-FF, T-FF, JK-FF, and Latch
- Includes a programmable filter to remove AOI output glitch
- Specifies that all logics are synchronous in bus clk domain
- Includes a memory-mapped chip connected to the slave peripheral (IPS) bus. Programming model is organized per channel for a simplified software.

25.3 Functional description

The following sections describe functional details of EVTG.

NOTE

n in EVTG*n* denotes the number of EVTG blocks.

25.3.1 Input sync and filter logic description

EVTG supports sync to EVTG inputs and filter function to AOI output.

Both features are disabled by default. To enable input sync function, configure EVTG*n*_CTRL[SYNC_CTRL], or to enable AOI filter function, configure a non-zero value to EVTGn_AOIm_FILT[FITL_PER].

The application scenario for both is different. The input sync feature makes EVTG input sync for two bus clock cycles. Remove the glitch whose width is less than a bus clock period. Because the Filter_Delay is "[(FILT_CNT + 3) x FILT_PER + 2]*(period of bus_clk)", any signal whose width is less than this value is filtered.

NXP recommends enabling just one among both features, and which one depends on the use case.

You can also enable both functions. For some use cases in Bypass mode and RS Trigger mode, both functions have to be disabled. Except for this scenario, NXP recommends enabling at least one input sync and filter.

25.3.2 Flip-flop modes

EVTG inside implements different kinds of flip-flops to generate the desired EVTG output. You can configure the flip-flop as Bypass mode, RS Trigger mode, T-FF mode, D-FF mode, JK-FF mode, and Latch mode.

25.3.2.1 Bypass mode

Bypass mode is selected when you configure EVTG*n*_CTRL[MODE_SEL] as 0h. So Bypass mode is used as a default mode.

In this mode:

- Flip-flop is passed, and directly assigns the two AOI expressions "AOI_0" and "AOI_1" to EVTG outputs (EVTG_OUTA and EVTG_OUTB).
- You can enable or disable input sync logic and filter function.

The following diagram shows the disabled case (input sync and filter are removed from the diagram).

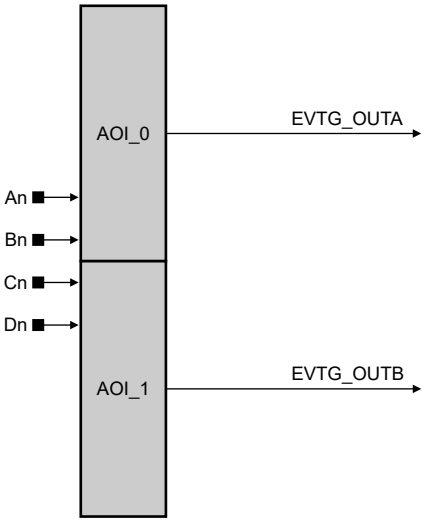


Figure 113. Bypass mode

25.3.2.2 RS Trigger mode

When you configure EVTG*n*_CTRL[MODE_SEL] as 1h, RS Trigger mode is selected.

In this mode the AOI_0 expression is Reset port, and AOI_1 is Set port. Both are active high. When R (Reset) is high, whatever S (Set) is, EVTG_OUTA = 0. When R is low, and S is high, EVTG_OUTA = 1. If both R and S are low, EVTG output is retained (see Figure 114).

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, you can enable or disable the input sync logic and filter function. The following diagram shows the disabled case (input sync and filter are removed from the following diagram).

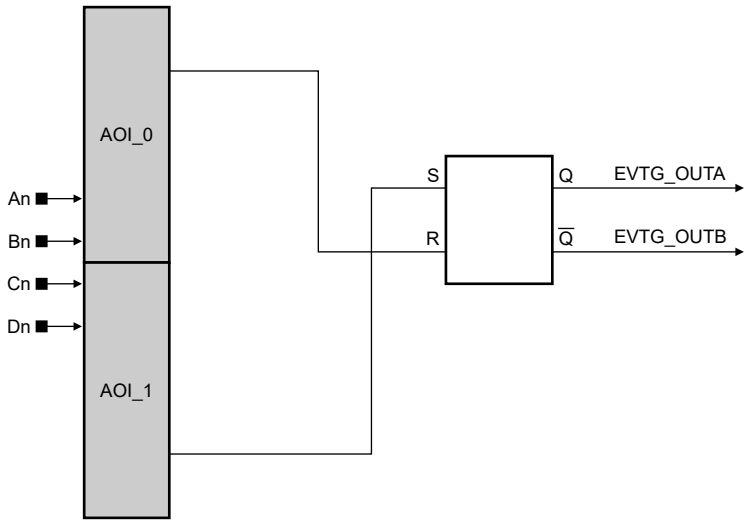


Figure 114. RS Trigger mode

25.3.2.3 T-FF mode

T-FF mode is selected when you configure EVTG*n*_CTRL[MODE_SEL] as 2h.

In this mode, the AOI_0 expression is T port of T-FF, and the AOI_1 is CLK port. When T asserts, the Q port (EVTG_OUTA) turnovers at the rising edge of CLK. When T de-asserts, Q (EVTG_OUTA) is retained (see Figure 115).

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, you must enable input sync or filter to remove the possible glitch.

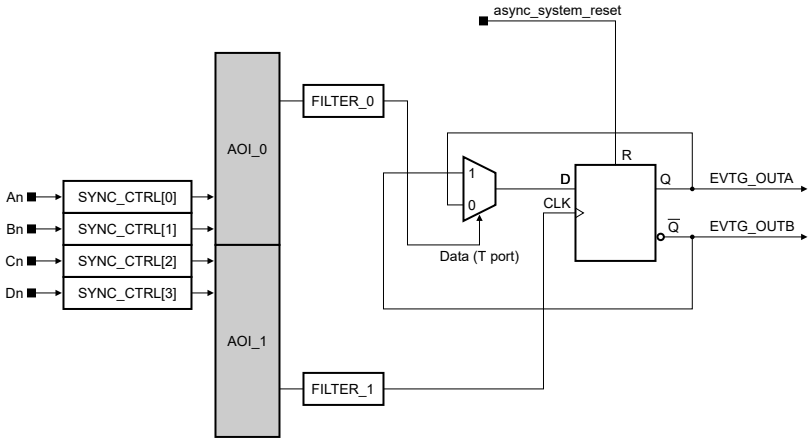


Figure 115. T-FF mode

25.3.2.4 D-FF mode

D-FF mode is selected when you configure EVTG*n*_CTRL[MODE_SEL] as 3h.

In this mode, the AOI_0 expression is the D port of D-FF and the AOI_1 expression is CLK port. At the rising edge of CLK, capture D to Q (EVTG_OUTA) (see Figure 116).

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, you must enable input sync or filter to remove the possible glitch.

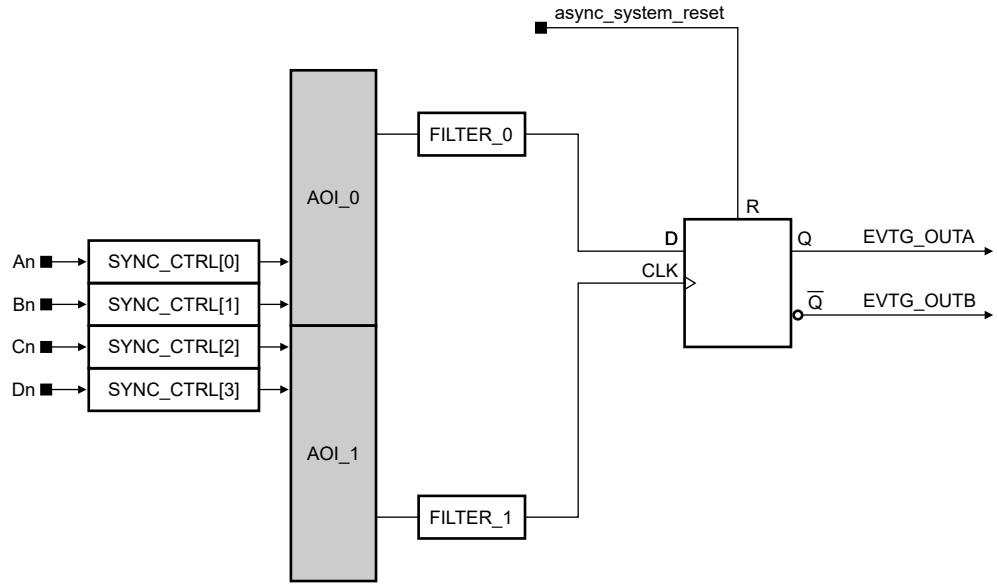


Figure 116. D-FF mode

25.3.2.5 JK-FF mode

In general, JK flip-flop has four input ports: J, K, Q, and CLK (Q is the output of the flip-flop), and the logical expression is "J&~Q | ~K&Q."

JK-FF mode is selected when you configure EVTG*n*_CTRL[MODE_SEL] as 4h.

Here you can implement the logic expression by AOI so that you can reuse the D-FF to implement JK-FF. Consider the following sample EVTG configuration, which Figure 117 also shows:

- An is a J port.
- Cn is a K port.
- Dn is a CLK port.
- Q port is FF feedback and override Bn.

According to the JK logic expression, the AOI_0 expression is "An&~Bn | Bn&~Cn," and the AOI_1 expression is "Dn." So the corresponding register configuration should be:

EVTGn_AOI0_BFT01 = 01101111_11011011;

EVTGn_AOI0_BFT23 = 00000000_00000000;

EVTGn_AOI1_BFT01 = 11111101_00000000;

EVTGn_AOI1_BFT23 = 00000000_00000000;

EVTGn_CTRL = 00001111_01010000; (EVTG*n*_CTRL[FB_OVRD] = 01)

EVTG*n*_CTRL[FB_OVRD] represents, which EVTG input the FF output (EVTG_OUTA) replaces. In this case, Bn is replaced, this is why EVTG*n*_CTRL[FB_OVRD] = 01.

In this mode, you must enable input sync or filter to remove the possible glitch.

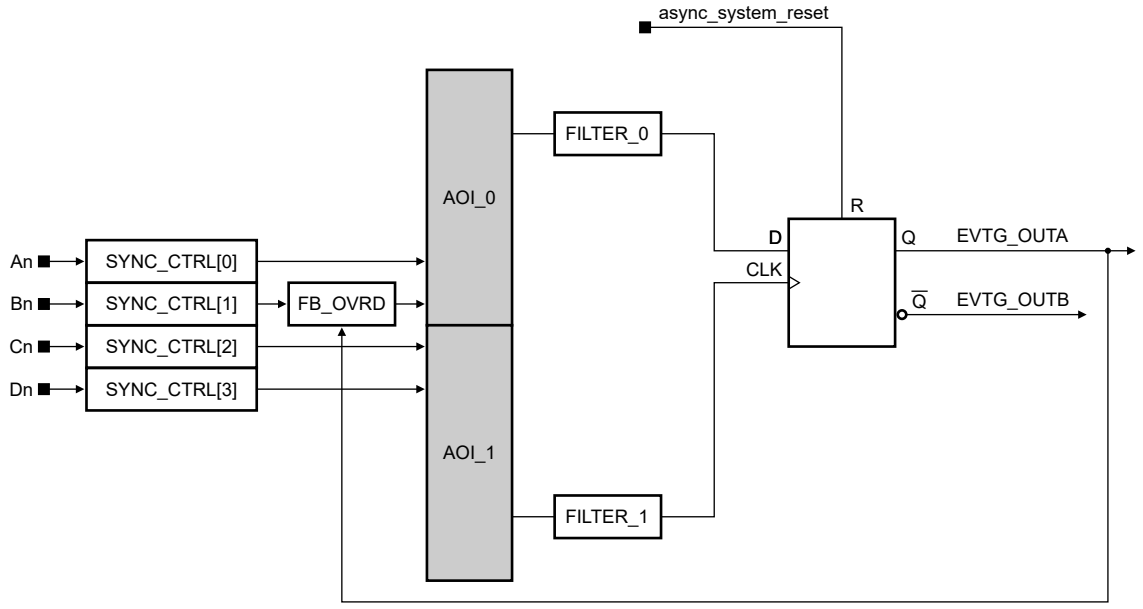


Figure 117. JK-FF mode

25.3.2.6 Latch mode

Latch mode is selected when you configure EVTG*n*_CTRL[MODE_SEL] as 5h.

In this mode, the AOI_0 expression is D port, and AOI_1 is CLK port. Different from D-FF mode, in Latch mode, D port is passed only when CLK is high, and output is retained when CLK is low.

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, you must enable input sync or filter to remove the possible glitch.

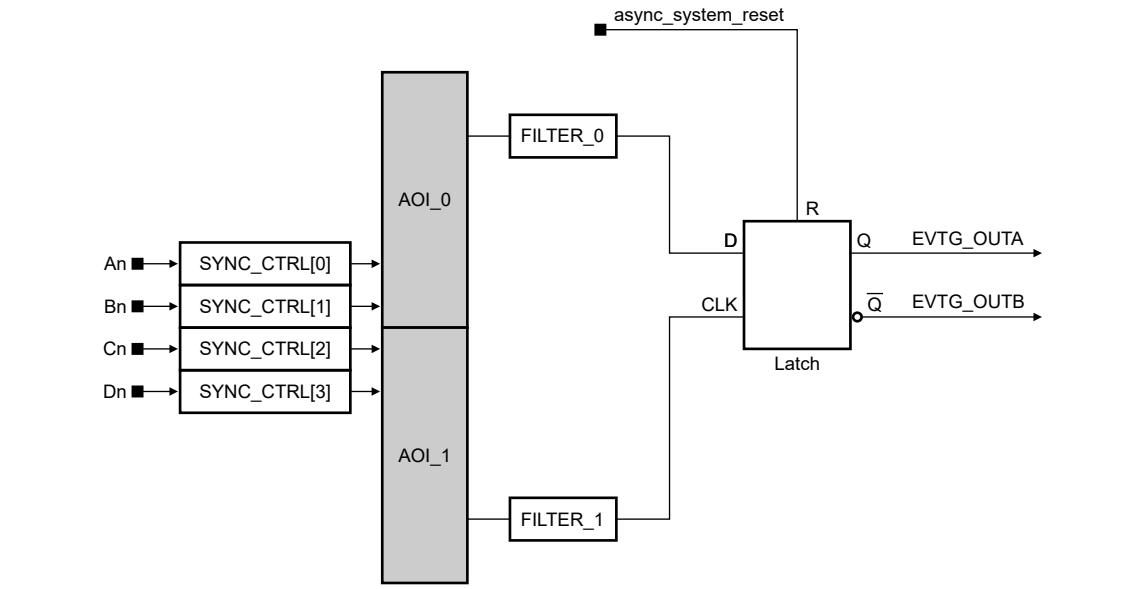


Figure 118. Latch mode

25.3.3 EVTG timing between inputs and outputs

- For Bypass mode and RS Trigger mode with input sync and filter disabled, there is no delay between inputs and outputs.
- For the modes with input sync and filter function enabled, the delay between input and output is:

  Sync_Delay: 2

  Flip-Flop_Delay: 1

  Filter_Delay: (FILT_CNT + 3) x FILT_PER + 2

  Whole_Delay = ((FILT_CNT + 3) x FILT_PER + 2) + 3

  In this case, all signals are synchronous in bus clk.

25.3.4 Clocks

EVTG includes only bus clock.

25.4 External signals

EVTG includes no external I/O signals.

25.5 Application information

This section describes applications that EVTG supports.

25.5.1 Configuration examples for AOI combinational function

This section describes the examples of the programming model configuration for simple boolean expressions.

AOI module provides a universal boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (A, B, C, D). Specifically, the EVTG output is defined by the following “4 x 4” boolean expression:

EVTGn_AOIm(m =0,1)

= (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0

| (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1

| (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2

| (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3

You can configure each selected input term in each product term to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product term uses eight bits of configuration information, two bits for each of the four selected event inputs. The actual boolean expression implemented in each channel is:

EVTGn_AOIm(m=0,1)

= (PT0_AC[0] & A | PT0_AC[1] & ~A)// product term 0

& (PT0_BC[0] & B | PTO_BC[1] & ~B)

& (PT0_CC[0] & C | PTO_CC[1] & ~C)

& (PT0_DC[0] & D | PTO_DC[1] & ~D)

| (PT1_AC[0] & A | PT1_AC[1] & ~A)// product term 1

& (PT1_BC[0] & B | PT1_BC[1] & ~B)

& (PT1_CC[0] & C | PT1_CC[1] & ~C)

& (PT1_DC[0] & D | PT1_DC[1] & ~D)

| (PT2_AC[0] & A | PT2_AC[1] & ~A)// product term 2

& (PT2_BC[0] & B | PT2_BC[1] & ~B)

& (PT2_CC[0] & C | PT2_CC[1] & ~C)

& (PT2_DC[0] & D | PT2_DC[1] & ~D)

| (PT3_AC[0] & A | PT3_AC[1] & ~A)// product term 3

& (PT3_BC[0] & B | PT3_BC[1] & ~B)

& (PT3_CC[0] & C | PT3_CC[1] & ~C)

& (PT3_DC[0] & D | PT3_DC[1] & ~D)

The following table shows the settings of the 32-bit registers for several simple boolean expressions.

Table 306. EVTGn_AOIm_BFT(m=0,1) values for simple boolean expressions

Event output expression	PT0	PT1	PT2	PT3	{EVTGn_AOIm_BFT01, EVTGn_AOIm_BFT23}
A&B	A&B	0	0	0	01011111_00000000_00000000_00000000
A&B&C	A&B&C	0	0	0	01010111_00000000_00000000_00000000
(A&B&C) | D	A&B&C	D	0	0	01010111_11111101_00000000_00000000
A|B|C|D	A	B	C	D	01111111_11011111_11110111_11111101
(A & ~B) | (~A & B)	A & ~B	~A & B	0	0	01101111_10011111_00000000_00000000

These examples show that the resulting logic provides a simple yet powerful boolean function evaluation for defining an AOI output.

25.6 Memory map and register definition

According to the DSC core, EVTG is also word addressed, where a word is 16 bits. Functionality for accesses of other widths is undefined.

25.6.1 EVTG register descriptions

25.6.1.1 EVTG memory map

EVTG0 base address: 400D_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	AOI0 Boolean Function Term 0 and 1 Configuration (EVTG0_AOI0_BFT01)	16	RW	0000h
2h	AOI0 Boolean Function Term 2 and 3 Configuration (EVTG0_AOI0_BFT23)	16	RW	0000h
4h	AOI1 Boolean Function Term 0 and 1 Configuration (EVTG0_AOI1_BFT01)	16	RW	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
6h	AOI1 Boolean Function Term 2 and 3 Configuration (EVTG0_AOI1_BFT23)	16	RW	0000h
Ah	Control and Status (EVTG0_CTRL)	16	RW	0000h
Ch	AOI0 Output Filter (EVTG0_AOI0_FILT)	16	RW	0000h
Eh	AOI1 Output Filter (EVTG0_AOI1_FILT)	16	RW	0000h
10h	AOI0 Boolean Function Term 0 and 1 Configuration (EVTG1_AOI0_BFT01)	16	RW	0000h
12h	AOI0 Boolean Function Term 2 and 3 Configuration (EVTG1_AOI0_BFT23)	16	RW	0000h
14h	AOI1 Boolean Function Term 0 and 1 Configuration (EVTG1_AOI1_BFT01)	16	RW	0000h
16h	AOI1 Boolean Function Term 2 and 3 Configuration (EVTG1_AOI1_BFT23)	16	RW	0000h
1Ah	Control and Status (EVTG1_CTRL)	16	RW	0000h
1Ch	AOI0 Output Filter (EVTG1_AOI0_FILT)	16	RW	0000h
1Eh	AOI1 Output Filter (EVTG1_AOI1_FILT)	16	RW	0000h
20h	AOI0 Boolean Function Term 0 and 1 Configuration (EVTG2_AOI0_BFT01)	16	RW	0000h
22h	AOI0 Boolean Function Term 2 and 3 Configuration (EVTG2_AOI0_BFT23)	16	RW	0000h
24h	AOI1 Boolean Function Term 0 and 1 Configuration (EVTG2_AOI1_BFT01)	16	RW	0000h
26h	AOI1 Boolean Function Term 2 and 3 Configuration (EVTG2_AOI1_BFT23)	16	RW	0000h
2Ah	Control and Status (EVTG2_CTRL)	16	RW	0000h
2Ch	AOI0 Output Filter (EVTG2_AOI0_FILT)	16	RW	0000h
2Eh	AOI1 Output Filter (EVTG2_AOI1_FILT)	16	RW	0000h
30h	AOI0 Boolean Function Term 0 and 1 Configuration (EVTG3_AOI0_BFT01)	16	RW	0000h
32h	AOI0 Boolean Function Term 2 and 3 Configuration (EVTG3_AOI0_BFT23)	16	RW	0000h
34h	AOI1 Boolean Function Term 0 and 1 Configuration (EVTG3_AOI1_BFT01)	16	RW	0000h
36h	AOI1 Boolean Function Term 2 and 3 Configuration (EVTG3_AOI1_BFT23)	16	RW	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
3Ah	Control and Status (EVTG3_CTRL)	16	RW	0000h
3Ch	AOI0 Output Filter (EVTG3_AOI0_FILT)	16	RW	0000h
3Eh	AOI1 Output Filter (EVTG3_AOI1_FILT)	16	RW	0000h

25.6.1.2 AOI0 Boolean Function Term 0 and 1 Configuration (EVTG0_AOI0_BFT01 - EVTG3_AOI0_BFT01)

Offset

Register	Offset
EVTG0_AOI0_BFT01	0h
EVTG1_AOI0_BFT01	10h
EVTG2_AOI0_BFT01	20h
EVTG3_AOI0_BFT01	30h

Function

Defines AOI0 boolean function selection for term 0 and term 1.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT0_AC		PT0_BC		PT0_CC		PT0_DC		PT1_AC		PT1_BC		PT1_CC		PT1_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 PT0_AC	Product Term 0, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 0. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT0_BC	Product Term 0, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT0_CC	Product Term 0, C Input Configuration Defines the boolean evaluation associated with the selected input C in product term 0. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT0_DC	Product Term 0, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 0. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT1_AC	Product Term 1, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 1. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT1_BC	Product Term 1, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 1. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT1_CC	Product Term 1, C Input Configuration Defines the boolean evaluation associated with the selected input C in product term 1. 00b - Force the C input in this product term to a logical zero

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT1_DC	Product Term 1, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 1. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

25.6.1.3 AOI0 Boolean Function Term 2 and 3 Configuration (EVTG0_AOI0_BFT23 - EVTG3_AOI0_BFT23)

Offset

Register	Offset
EVTG0_AOI0_BFT23	2h
EVTG1_AOI0_BFT23	12h
EVTG2_AOI0_BFT23	22h
EVTG3_AOI0_BFT23	32h

Function

Defines the AOI0 boolean function selection for term 2 and term 3.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT2_AC		PT2_BC		PT2_CC		PT2_DC		PT3_AC		PT3_BC		PT3_CC		PT3_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 PT2_AC	Product Term 2, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT2_BC	Product Term 2, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 2. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT2_CC	Product Term 2, C Input Configuration Defines the boolean evaluation associated with the selected input C in product term 2. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT2_DC	Product Term 2, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 2. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT3_AC	Product Term 3, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 3. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT3_BC	Product Term 3, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 3. 00b - Force the B input in this product term to a logical zero

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT3_CC	Product Term 3, C Input Configuration Defines the boolean evaluation associated with the selected input C in product term 3. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT3_DC	Product Term 3, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 3. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

25.6.1.4 AOI1 Boolean Function Term 0 and 1 Configuration (EVTG0_AOI1_BFT01 - EVTG3_AOI1_BFT01)

Offset

Register	Offset
EVTG0_AOI1_BFT01	4h
EVTG1_AOI1_BFT01	14h
EVTG2_AOI1_BFT01	24h
EVTG3_AOI1_BFT01	34h

Function
Defines the AOI1 boolean function selection for term 0 and term 1.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT0_AC		PT0_BC		PT0_CC		PT0_DC		PT1_AC		PT1_BC		PT1_CC		PT1_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 PT0_AC	Product Term 0, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 0. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT0_BC	Product Term 0, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 0. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT0_CC	Product Term 0, C Input Configuration Defines the boolean evaluation associated with the selected input C in product term 0. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT0_DC	Product Term 0, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 0. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT1_AC	Product Term 1, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT1_BC	Product Term 1, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 1. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT1_CC	Product Term 1, C Input Configuration Defines the Boolean evaluation associated with the selected input C in product term 1. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT1_DC	Product Term 1, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 1. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

25.6.1.5 AOI1 Boolean Function Term 2 and 3 Configuration (EVTG0_AOI1_BFT23 - EVTG3_AOI1_BFT23)

Offset

Register	Offset
EVTG0_AOI1_BFT23	6h
EVTG1_AOI1_BFT23	16h
EVTG2_AOI1_BFT23	26h
EVTG3_AOI1_BFT23	36h

Function

Defines the AOI1 boolean function selection for term 2 and term 3.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT2_AC		PT2_BC		PT2_CC		PT2_DC		PT3_AC		PT3_BC		PT3_CC		PT3_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 PT2_AC	Product Term 2, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 2. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT2_BC	Product Term 2, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 2. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT2_CC	Product Term 2, C Input Configuration Defines the boolean evaluation associated with the selected input C in product term 2. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT2_DC	Product Term 2, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 2. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT3_AC	Product Term 3, A Input Configuration Defines the boolean evaluation associated with the selected input A in product term 3.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT3_BC	Product Term 3, B Input Configuration Defines the boolean evaluation associated with the selected input B in product term 3. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT3_CC	Product Term 3, C Input Configuration Defines the boolean evaluation associated with the selected input C in product term 3. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT3_DC	Product Term 3, D Input Configuration Defines the boolean evaluation associated with the selected input D in product term 3. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

25.6.1.6 Control and Status (EVTG0_CTRL - EVTG3_CTRL)

Offset

Register	Offset
EVTG0_CTRL	Ah
EVTG1_CTRL	1Ah
EVTG2_CTRL	2Ah
EVTG3_CTRL	3Ah

Function
Contains control and status register for EVTG.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		FORCE_BYPASS		SYNC_CTRL				FB_OVRD		Reserved	MODE_SEL			0	FF_INIT
W															INIT_EN	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 —	Reserved
13-12 FORCE_BYPASS	Force Bypass Control When MODE_SEL is set JK-FF mode, we should not enable FORCE_BYPASS, becasue the FB_OVRD will affect output 0xb - Will not force the bypass 1xb - Whatever MODE_SEL is, will force bypass flip-flop and route the AOI_1(Filter_1) value directly to EVTG_OUTB x0b - Will not force the bypass x1b - Whatever MODE_SEL is, will force bypass flip-flop and route the AOI_0(Filter_0) value directly to EVTG_OUTA
11-8 SYNC_CTRL	Synchronize Control Specifies EVTG inputs synchronous with bus clk. 0xxxb - EVTG input "Dn" will not be synced 1xxxb - EVTG input "Dn" will be synced by two bus clk cycles x0xxb - EVTG input "Cn" will not be synced x1xxb - EVTG input "Cn" will be synced by two bus clk cycles xx0xb - EVTG input "Bn" will not be synced xx1xb - EVTG input "Bn" will be synced by two bus clk cycles xxx0b - EVTG input "An" will not be synced xxx1b - EVTG input "An" will be synced by two bus clk cycles
7-6 FB_OVRD	EVTG Output Feedback Override Control Requires EVTG_OUTA feedback to EVTG input and replaces one of the four inputs, when EVTG*n*_CTRL[MODE_SEL] is configured as JK-FF mode. 00b - Replace An 01b - Replace Bn 10b - Replace Cn

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Replace Dn
5 —	Reserved
4-2 MODE_SEL	Flip-Flop Mode Selection Selects the flip-flop modes. 000b - Bypass mode 001b - RS Trigger mode 010b - T-FF mode 011b - D-FF mode 100b - JK-FF mode 101b - Latch mode 110b - Reserved 111b - Reserved
1 INIT_EN	Flip-Flop Initial Output Enable Control Forces the value of EVTG*n*_CTRL[FF_INIT] to present on flip-flop positive output. Write a 1 to this field to generate an enable pulse. Read this field returns 0. This field must become 1 after EVTG*n*_CTRL[FF_INIT] = 1. 0b - Write 0 does not generate enable pulse 1b - Write 1 generates enable pulse
0 FF_INIT	Flip flop Initial Value Configuration Configures the positive output of flip-flop either as 0 or 1. FF_INIT does not take effect until you write "1" into EVTG*n*_CTRL[INIT_EN]. This field must become 1 before EVTG*n*_CTRL[INIT_EN] = 1. 0b - 0 1b - 1

25.6.1.7 AOI0 Output Filter (EVTG0_AOI0_FILT - EVTG3_AOI0_FILT)

Offset

Register	Offset
EVTG0_AOI0_FILT	Ch
EVTG1_AOI0_FILT	1Ch

Table continues on the next page...

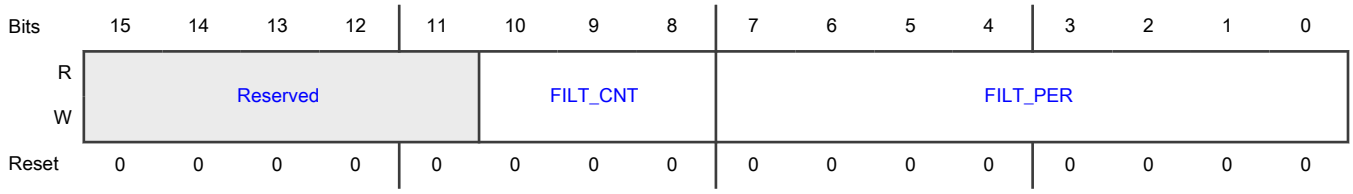
Table continued from the previous page...

Register	Offset
EVTG2_AOI0_FILT	2Ch
EVTG3_AOI0_FILT	3Ch

Function

Contains AOI0 output filter control register.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-8 FILT_CNT	Output Filter Sample Count Represents the number of consecutive samples that must agree before the output filter accepts an input transition. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Output Filter Sample Period Represents the input signals' sampling period (in IP bus clock cycles). Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 00h (default), then the output filter is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write zero first to clear the filter.

25.6.1.8 AOI1 Output Filter (EVTG0_AOI1_FILT - EVTG3_AOI1_FILT)

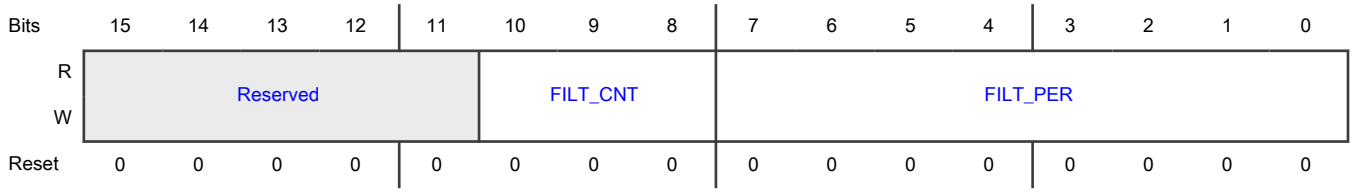
Offset

Register	Offset
EVTG0_AOI1_FILT	Eh
EVTG1_AOI1_FILT	1Eh
EVTG2_AOI1_FILT	2Eh
EVTG3_AOI1_FILT	3Eh

Function

Contains AOI1 output filter control register.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-8 FILT_CNT	Output Filter Sample Count Represents the number of consecutive samples that must agree before the output filter accepts an output transition. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Output Filter Sample Period Represents the output signals' sampling period (in IP bus clock cycles). Each output is sampled multiple times at the rate which this field specifies. The output filter is bypassed if FILT_PER is 00h (default). The value of FILT_PER affects the output latency. When you change the values of this field from one non-zero to another non-zero value, first write zero to clear the filter.

Chapter 26
Input Multiplexing (INPUTMUX)

26.1 Chip-specific INPUTMUX information

Table 307. Reference links to related information

Topic	Related module	Reference
Full description	INPUTMUX	INPUTMUX
Peripheral memory map		PBRG memory map

Once set up, no clocks are required for the input multiplexer to function. The system clock is needed only to write to or read from the INPUTMUX registers. Once the input multiplexer is configured, disable the clock to the INPUTMUX module in the AHBCLKCTRL register.

26.1.1 Module instances

This device contains one instance of the INPUTMUX module, INPUTMUX0.

26.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

26.2 Overview

The Input Multiplexing module (INPUTMUX) provides signal routing options for internal peripherals. Some peripheral inputs are multiplexed to multiple input sources. The sources can be external pins, interrupts, output signals of other peripherals, or other internal signals.

NOTE

Depending on the package, not all inputs from external pins may be available.

26.2.1 Block Diagram

Figure 119 shows a generic input multiplexer arrangement with n inputs.

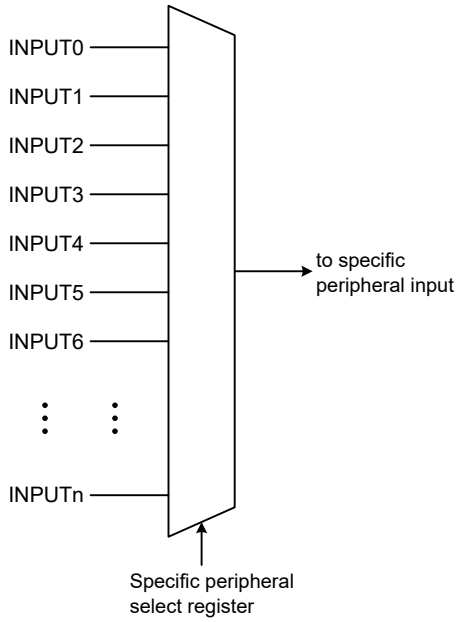


Figure 119. Generic input multiplexing

26.2.2 Features

- Configures the inputs to the SCTIMER.
- Configures the inputs to the asynchronous CTimers.
- Configures the inputs to the Pin Interrupt and Pattern Match (PINT).
- Configures the inputs to CMP, ADC, DAC, QDC, PWM, PWM EXT clock, AOI, TRIG_OUT pins.
- Configures the inputs to the Frequency Measurement (FREQME).

26.3 Functional description

The INPUTMUX implements a number of input multiplexers that select one of many inputs to be routed to a specific input signal for a given peripheral. This is used to allow user configuration of data paths between internal modules and/or external pins on the device. For every module input (output from the INPUTMUX), there is a register that selects the input to use, where the register name and description provide details on the module input controlled by each register. The input signal/pin options for each of the muxes are configurable, and can vary from mux to mux. Refer to the register descriptions for the details on the input signal/pin options used for each INPUTMUX output in Memory map and register definition.

26.4 External signals

The INPUTMUX has no dedicated pins. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

26.5 Memory map and register definition

This section includes the INPUTMUX module memory map and detailed descriptions of all registers.

26.5.1 INPUTMUX register descriptions

26.5.1.1 INPUTMUX memory map

INPUTMUX0 base address: 4000_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 1Ch	Inputmux Register for SCT0 Input (SCT0_INMUX0 - SCT0_INMUX7)	32	RW	0000_007Fh
20h	Capture Select Register for CTIMER Inputs (CTIMER0CAP0)	32	RW	0000_007Fh
24h	Capture Select Register for CTIMER Inputs (CTIMER0CAP1)	32	RW	0000_007Fh
28h	Capture Select Register for CTIMER Inputs (CTIMER0CAP2)	32	RW	0000_007Fh
2Ch	Capture Select Register for CTIMER Inputs (CTIMER0CAP3)	32	RW	0000_007Fh
30h	Trigger Register for CTIMER (TIMER0TRIG)	32	RW	0000_007Fh
40h	Capture Select Register for CTIMER Inputs (CTIMER1CAP0)	32	RW	0000_007Fh
44h	Capture Select Register for CTIMER Inputs (CTIMER1CAP1)	32	RW	0000_007Fh
48h	Capture Select Register for CTIMER Inputs (CTIMER1CAP2)	32	RW	0000_007Fh
4Ch	Capture Select Register for CTIMER Inputs (CTIMER1CAP3)	32	RW	0000_007Fh
50h	Trigger Register for CTIMER (TIMER1TRIG)	32	RW	0000_007Fh
60h	Capture Select Register for CTIMER Inputs (CTIMER2CAP0)	32	RW	0000_007Fh
64h	Capture Select Register for CTIMER Inputs (CTIMER2CAP1)	32	RW	0000_007Fh
68h	Capture Select Register for CTIMER Inputs (CTIMER2CAP2)	32	RW	0000_007Fh
6Ch	Capture Select Register for CTIMER Inputs (CTIMER2CAP3)	32	RW	0000_007Fh
70h	Trigger Register for CTIMER (TIMER2TRIG)	32	RW	0000_007Fh
A0h - BCh	Inputmux Register for SMARTDMA Arch B Inputs (SMARTDMAARCHB_INMUX0 - SMARTDMAARCHB_INMUX7)	32	RW	0000_007Fh
C0h - DCh	Pin Interrupt Select (PINTSEL0 - PINTSEL7)	32	RW	0000_007Fh
180h	Selection for Frequency Measurement Reference Clock (FREQMEAS_REF)	32	RW	0000_003Fh
184h	Selection for Frequency Measurement Target Clock (FREQMEAS_TAR)	32	RW	0000_003Fh
1A0h	Capture Select Register for CTIMER Inputs (CTIMER3CAP0)	32	RW	0000_007Fh
1A4h	Capture Select Register for CTIMER Inputs (CTIMER3CAP1)	32	RW	0000_007Fh
1A8h	Capture Select Register for CTIMER Inputs (CTIMER3CAP2)	32	RW	0000_007Fh
1ACh	Capture Select Register for CTIMER Inputs (CTIMER3CAP3)	32	RW	0000_007Fh
1B0h	Trigger Register for CTIMER (TIMER3TRIG)	32	RW	0000_007Fh
1C0h	Capture Select Register for CTIMER Inputs (CTIMER4CAP0)	32	RW	0000_007Fh
1C4h	Capture Select Register for CTIMER Inputs (CTIMER4CAP1)	32	RW	0000_007Fh
1C8h	Capture Select Register for CTIMER Inputs (CTIMER4CAP2)	32	RW	0000_007Fh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
1CCh	Capture Select Register for CTIMER Inputs (CTIMER4CAP3)	32	RW	0000_007Fh
1D0h	Trigger Register for CTIMER (TIMER4TRIG)	32	RW	0000_007Fh
260h	CMP0 Input Connections (CMP0_TRIG)	32	RW	0000_003Fh
280h - 28Ch	ADC Trigger Input Connections (ADC0_TRIG0 - ADC0_TRIG3)	32	RW	0000_007Fh
2C0h - 2CCh	ADC Trigger Input Connections (ADC1_TRIG0 - ADC1_TRIG3)	32	RW	0000_007Fh
300h	DAC0 Trigger Inputs (DAC0_TRIG)	32	RW	0000_003Fh
320h	DAC1 Trigger Inputs (DAC1_TRIG)	32	RW	0000_003Fh
340h	DAC2 Trigger Inputs (DAC2_TRIG)	32	RW	0000_003Fh
360h	QDC0 Trigger Input Connections (QDC0_TRIG)	32	RW	0000_003Fh
364h	QDC0 Input Connections (QDC0_HOME)	32	RW	0000_003Fh
368h	QDC0 Input Connections (QDC0_INDEX)	32	RW	0000_003Fh
36Ch	QDC0 Input Connections (QDC0_PHASEB)	32	RW	0000_003Fh
370h	QDC0 Input Connections (QDC0_PHASEA)	32	RW	0000_003Fh
380h	QDC1 Trigger Input Connections (QDC1_TRIG)	32	RW	0000_003Fh
384h	QDC1 Input Connections (QDC1_HOME)	32	RW	0000_003Fh
388h	QDC1 Input Connections (QDC1_INDEX)	32	RW	0000_003Fh
38Ch	QDC1 Input Connections (QDC1_PHASEB)	32	RW	0000_003Fh
390h	QDC1 Input Connections (QDC1_PHASEA)	32	RW	0000_003Fh
3A0h - 3ACh	PWM0 External Synchronization (FlexPWM0_SM0_EXTSYNC - FlexPWM0_SM3_EXTSYNC)	32	RW	0000_003Fh
3B0h - 3BCh	PWM0 Input Trigger Connections (FlexPWM0_SM0_EXTA - FlexPWM0_SM3_EXTA)	32	RW	0000_003Fh
3C0h	PWM0 External Force Trigger Connections (FlexPWM0_EXTFORCE)	32	RW	0000_003Fh
3C4h - 3D0h	PWM0 Fault Input Trigger Connections (FlexPWM0_FAULT0 - FlexPWM0_FAULT3)	32	RW	0000_003Fh
3E0h - 3ECh	PWM1 External Synchronization (FlexPWM1_SM0_EXTSYNC - FlexPWM1_SM3_EXTSYNC)	32	RW	0000_003Fh
3F0h - 3FCh	PWM1 Input EXTA Connections (FlexPWM1_SM0_EXTA - FlexPWM1_SM3_EXTA)	32	RW	0000_003Fh
400h	PWM1 External Force Trigger Connections (FlexPWM1_EXTFORCE)	32	RW	0000_003Fh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
404h - 410h	PWM1 Fault Input Trigger Connections (FlexPWM1_FAULT0 - FlexPWM1_FAULT3)	32	RW	0000_003Fh
420h	PWM0 External Clock Trigger (PWM0_EXT_CLK)	32	RW	0000_0007h
424h	PWM1 External Clock Trigger (PWM1_EXT_CLK)	32	RW	0000_0007h
440h - 47Ch	EVTG Trigger Input Connections (EVTG_TRIG0 - EVTG_TRIG15)	32	RW	0000_003Fh
480h	USB-FS Trigger Input Connections (USBFS_TRIG)	32	RW	0000_000Fh
4A0h	TSI Trigger Input Connections (TSI_TRIG)	32	RW	0000_0003h
4C0h - 4DCh	EXT Trigger Connections (EXT_TRIG0 - EXT_TRIG7)	32	RW	0000_003Fh
4E0h	CMP1 Input Connections (CMP1_TRIG)	32	RW	0000_003Fh
500h	CMP2 Input Connections (CMP2_TRIG)	32	RW	0000_003Fh
520h - 530h	SINC Filter Channel Trigger Input Connections (SINC_FILTER_CH0 - SINC_FILTER_CH4)	32	RW	0000_003Fh
580h - 588h	OPAMP Trigger Input Connections (OPAMP0_TRIG - OPAMP2_TRIG)	32	RW	0000_003Fh
5A0h	LP_FLEXCOMM0 Trigger Input Connections (FLEXCOMM0_TRIG)	32	RW	0000_003Fh
5C0h	LP_FLEXCOMM1 Trigger Input Connections (FLEXCOMM1_TRIG)	32	RW	0000_003Fh
5E0h	LP_FLEXCOMM2 Trigger Input Connections (FLEXCOMM2_TRIG)	32	RW	0000_003Fh
600h	LP_FLEXCOMM3 Trigger Input Connections (FLEXCOMM3_TRIG)	32	RW	0000_003Fh
620h	LP_FLEXCOMM4 Trigger Input Connections (FLEXCOMM4_TRIG)	32	RW	0000_003Fh
640h	LP_FLEXCOMM5 Trigger Input Connections (FLEXCOMM5_TRIG)	32	RW	0000_003Fh
660h	LP_FLEXCOMM6 Trigger Input Connections (FLEXCOMM6_TRIG)	32	RW	0000_003Fh
680h	LP_FLEXCOMM7 Trigger Input Connections (FLEXCOMM7_TRIG)	32	RW	0000_003Fh
6A0h	LP_FLEXCOMM8 Trigger Input Connections (FLEXCOMM8_TRIG)	32	RW	0000_003Fh
6C0h	LP_FLEXCOMM9 Trigger Input Connections (FLEXCOMM9_TRIG)	32	RW	0000_003Fh
6E0h - 6FCh	FlexIO Trigger Input Connections (FLEXIO_TRIG0 - FLEXIO_TRIG7)	32	RW	0000_007Fh
700h	DMA0 Request Enable0 (DMA0_REQ_ENABLE0)	32	RW	FFFF_FFFFh
704h	DMA0 Request Enable0 (DMA0_REQ_ENABLE0_SET)	32	RW	0000_0000h
708h	DMA0 Request Enable0 (DMA0_REQ_ENABLE0_CLR)	32	RW	0000_0000h
70Ch	DMA0 Request Enable0 (DMA0_REQ_ENABLE0_TOG)	32	RW	0000_0000h
710h	DMA0 Request Enable1 (DMA0_REQ_ENABLE1)	32	RW	FFFF_FFFFh
714h	DMA0 Request Enable1 (DMA0_REQ_ENABLE1_SET)	32	RW	0000_0000h
718h	DMA0 Request Enable1 (DMA0_REQ_ENABLE1_CLR)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
71Ch	DMA0 Request Enable1 (DMA0_REQ_ENABLE1_TOG)	32	RW	0000_0000h
720h	DMA0 Request Enable2 (DMA0_REQ_ENABLE2)	32	RW	FFFF_FFFFh
724h	DMA0 Request Enable2 (DMA0_REQ_ENABLE2_SET)	32	RW	0000_0000h
728h	DMA0 Request Enable2 (DMA0_REQ_ENABLE2_CLR)	32	RW	0000_0000h
72Ch	DMA0 Request Enable2 (DMA0_REQ_ENABLE2_TOG)	32	RW	0000_0000h
730h	DMA0 Request Enable3 (DMA0_REQ_ENABLE3)	32	RW	03FF_FFFFh
734h	DMA0 Request Enable3 (DMA0_REQ_ENABLE3_SET)	32	RW	0000_0000h
738h	DMA0 Request Enable3 (DMA0_REQ_ENABLE3_CLR)	32	RW	0000_0000h
780h	DMA1 Request Enable0 (DMA1_REQ_ENABLE0)	32	RW	FFFF_FFFFh
784h	DMA1 Request Enable0 (DMA1_REQ_ENABLE0_SET)	32	RW	0000_0000h
788h	DMA1 Request Enable0 (DMA1_REQ_ENABLE0_CLR)	32	RW	0000_0000h
78Ch	DMA1 Request Enable0 (DMA1_REQ_ENABLE0_TOG)	32	RW	0000_0000h
790h	DMA1 Request Enable1 (DMA1_REQ_ENABLE1)	32	RW	FFFF_FFFFh
794h	DMA1 Request Enable1 (DMA1_REQ_ENABLE1_SET)	32	RW	0000_0000h
798h	DMA1 Request Enable1 (DMA1_REQ_ENABLE1_CLR)	32	RW	0000_0000h
79Ch	DMA1 Request Enable1 (DMA1_REQ_ENABLE1_TOG)	32	RW	0000_0000h
7A0h	DMA1 Request Enable2 (DMA1_REQ_ENABLE2)	32	RW	FFFF_FFFFh
7A4h	DMA1 Request Enable2 (DMA1_REQ_ENABLE2_SET)	32	W	0000_0000h
7A8h	DMA1 Request Enable2 (DMA1_REQ_ENABLE2_CLR)	32	W	0000_0000h
7ACh	DMA1 Request Enable2 (DMA1_REQ_ENABLE2_TOG)	32	W	0000_0000h
7B0h	DMA1 Request Enable3 (DMA1_REQ_ENABLE3)	32	RW	03FF_FFFFh
7B4h	DMA1 Request Enable3 (DMA1_REQ_ENABLE3_SET)	32	RW	0000_0000h
7B8h	DMA1 Request Enable3 (DMA1_REQ_ENABLE3_CLR)	32	RW	0000_0000h

26.5.1.2 Inputmux Register for SCT0 Input (SCT0_INMUX0 - SCT0_INMUX7)

Offset

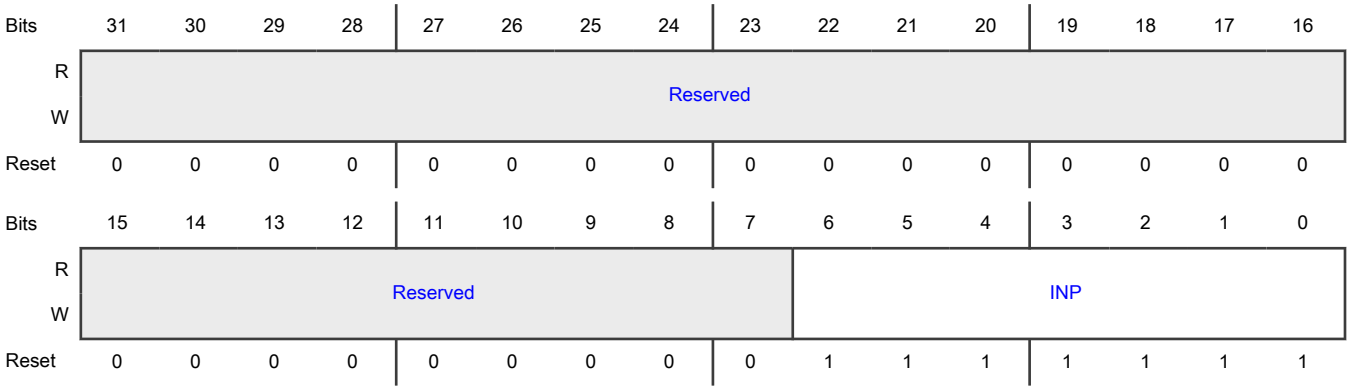
For a = 0 to 7:

Register	Offset
SCT0_INMUXa	0h + (a × 4h)

Function

This register selects one input source for each SCT0 input from 24 external and internal sources. (An exception is SCT0 input SCT0_INMUX7, which is directly connected to SCTASYNCCLK (output of SCT clock divider) PLL clock and not multiplexed with any other signals).The output of SCT0 Input multiplexing register 0 selects the source for SCT0 input 0. The output of SCT0 Input multiplexing register 1 selects the source for SCT0 input 1, and so forth up to SCT0 Input multiplexing register 6, which selects the input for SCT0 input 6.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 INP	Input number to SCT0 inputs. 000_0000b - SCT0_IN0 input is selected 000_0001b - SCT0_IN1 input is selected 000_0010b - SCT0_IN2 input is selected 000_0011b - SCT0_IN3 input is selected 000_0100b - SCT0_IN4 input is selected 000_0101b - SCT0_IN5 input is selected 000_0110b - SCT0_IN6 input is selected 000_0111b - SCT0_IN7 input is selected 000_1000b - CTIMER0_MAT0 input is selected 000_1001b - CTIMER1_MAT0 input is selected 000_1010b - CTIMER2_MAT0 input is selected 000_1011b - CTIMER3_MAT0 input is selected 000_1100b - CTIMER4_MAT0 input is selected 000_1101b - ADC0 ADC0_IRQ input is selected 000_1110b - PINT GPIO_INT_BMAT input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000_1111b - usb0 start of frame input is selected 001_0000b - usb1 start of frame input is selected 001_0001b - SINC Filter CH0 Conversion Complete input is selected 001_0010b - SINC Filter CH1 Conversion Complete input is selected 001_0011b - SINC Filter CH2 Conversion Complete input is selected 001_0100b - SINC Filter CH3 Conversion Complete input is selected 001_0101b - SINC Filter CH4 Conversion Complete input is selected 001_0110b - Reserved 001_0111b - DEBUG_HALTED input is selected 001_1000b - ADC1_IRQ input is selected 001_1001b - ADC0_tcomp[0] input is selected 001_1010b - ADC0_tcomp[1] input is selected 001_1011b - ADC0_tcomp[2] input is selected 001_1100b - ADC0_tcomp[3] input is selected 001_1101b - ADC1_tcomp[0] input is selected 001_1110b - ADC1_tcomp[1] input is selected 001_1111b - ADC1_tcomp[2] input is selected 010_0000b - ADC1_tcomp[3] input is selected 010_0001b - CMP0_OUT input is selected 010_0010b - CMP1_OUT input is selected 010_0011b - CMP2_OUT input is selected 010_0100b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 010_0101b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 010_0110b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 010_0111b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 010_1000b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 010_1001b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 010_1010b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 010_1011b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 010_1100b - QDC0_CMP/POS_MATCH input is selected 010_1101b - QDC1_CMP/POS_MATCH input is selected 010_1110b - EVTG_OUT0A input is selected 010_1111b - EVTG_OUT0B input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011_0000b - EVTG_OUT1A input is selected 011_0001b - EVTG_OUT1B input is selected 011_0010b - EVTG_OUT2A input is selected 011_0011b - EVTG_OUT2B input is selected 011_0100b - EVTG_OUT3A input is selected 011_0101b - EVTG_OUT3B input is selected 011_0110b - FC3_P0 (SDO, SDA) input is selected 011_0111b - FC3_P1 (SCK, TXD, SCL) input is selected 011_1000b - FC3_P2 (RTS, SCLS, TXD) input is selected 011_1001b - FC3_P3 (PCS[0], CTS, SDAS) input is selected 011_1010b - Reserved 011_1011b - Reserved 011_1100b - LP_FLEXCOMM0 trig 0 (lpuart_trg_txword) input is selected 011_1101b - LP_FLEXCOMM0 trig 1 (lpuart_trg_rxword) input is selected 011_1110b - LP_FLEXCOMM0 trig 2 (lpuart_trg_rxidle) input is selected 011_1111b - LP_FLEXCOMM1 trig 0 input is selected 100_0000b - LP_FLEXCOMM1 trig 1 input is selected 100_0001b - LP_FLEXCOMM1 trig 2 input is selected 100_0010b - LP_FLEXCOMM2 trig 0 input is selected 100_0011b - LP_FLEXCOMM2 trig 1 input is selected 100_0100b - LP_FLEXCOMM2 trig 2 input is selected 100_0101b - LP_FLEXCOMM3 trig 0 input is selected 100_0110b - LP_FLEXCOMM3 trig 1 input is selected 100_0111b - LP_FLEXCOMM3 trig 2 input is selected 100_1000b - LP_FLEXCOMM3 trig 3 input is selected 100_1001b - SAI0 TX BCLK input is selected 100_1010b - SAI0 RX BCLK input is selected 100_1011b - SAI1 TX BCLK input is selected 100_1100b - SAI1 RX BCLK input is selected All other values are reserved.

26.5.1.3 Capture Select Register for CTIMER Inputs (CTIMER0CAP0 - CTIMER2CAP3)

Offset

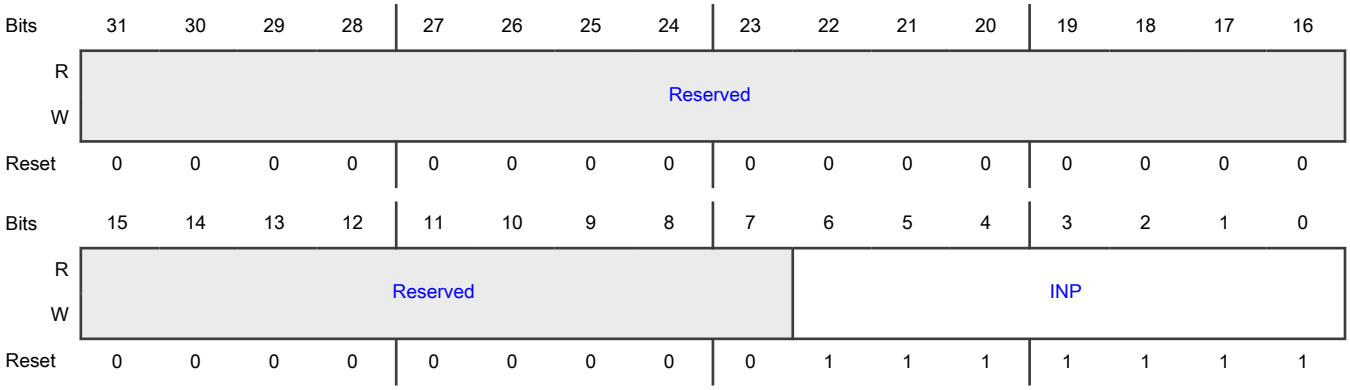
For a = 0 to 2; b = 0 to 3:

Register	Offset
CTIMERaCAPb	20h + (a × 20h) + (b × 4h)

Function

For each of the 5 standard timers, numbered i = 0 to 4 there are 4 CTIMERiCAPTj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources. The output of TIMER0CAPT0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPT1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPT3. Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 INP	Input number for CTIMER 000_0000b - CT_INP0 input is selected 000_0001b - CT_INP1 input is selected 000_0010b - CT_INP2 input is selected 000_0011b - CT_INP3 input is selected 000_0100b - CT_INP4 input is selected 000_0101b - CT_INP5 input is selected 000_0110b - CT_INP6 input is selected 000_0111b - CT_INP7 input is selected 000_1000b - CT_INP8 input is selected 000_1001b - CT_INP9 input is selected 000_1010b - CT_INP10 input is selected 000_1011b - CT_INP11 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000_1100b - CT_INP12 input is selected 000_1101b - CT_INP13 input is selected 000_1110b - CT_INP14 input is selected 000_1111b - CT_INP15 input is selected 001_0000b - CT_INP16 input is selected 001_0001b - CT_INP17 input is selected 001_0010b - CT_INP18 input is selected 001_0011b - CT_INP19 input is selected 001_0100b - usb0 start of frame input is selected 001_0101b - usb1 start of frame input is selected 001_0110b - DCDC_BURST_ACTIVE input is selected 001_0111b - sai0_tx_sync_out[1] input is selected 001_1000b - sai0_rx_sync_out[2] input is selected 001_1001b - ADC0_IRQ input is selected 001_1010b - ADC1_IRQ input is selected 001_1011b - CMP0_OUT input is selected 001_1100b - CMP1_OUT input is selected 001_1101b - CMP2_OUT input is selected 001_1110b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 001_1111b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 010_0000b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 010_0001b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 010_0010b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 010_0011b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 010_0100b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 010_0101b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 010_0110b - QDC0_CMP/POS_MATCH input is selected 010_0111b - QDC1_CMP/POS_MATCH input is selected 010_1000b - EVTG_OUT0A input is selected 010_1001b - EVTG_OUT0B input is selected 010_1010b - EVTG_OUT1A input is selected 010_1011b - EVTG_OUT1B input is selected 010_1100b - EVTG_OUT2A input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	010_1101b - EVTG_OUT2B input is selected 010_1110b - EVTG_OUT3A input is selected 010_1111b - EVTG_OUT3B input is selected 011_0000b - Reserved 011_0001b - Reserved 011_0010b - LP_FLEXCOMM0 trig 0 input is selected 011_0011b - LP_FLEXCOMM0 trig 1 input is selected 011_0100b - LP_FLEXCOMM0 trig 2 input is selected 011_0101b - LP_FLEXCOMM1 trig 0 input is selected 011_0110b - LP_FLEXCOMM1 trig 1 input is selected 011_0111b - LP_FLEXCOMM1 trig 2 input is selected 011_1000b - LP_FLEXCOMM2 trig 0 input is selected 011_1001b - LP_FLEXCOMM2 trig 1 input is selected 011_1010b - LP_FLEXCOMM2 trig 2 input is selected 011_1011b - LP_FLEXCOMM3 trig 0 input is selected 011_1100b - LP_FLEXCOMM3 trig 1 input is selected 011_1101b - LP_FLEXCOMM3 trig 2 input is selected 011_1110b - LP_FLEXCOMM3 trig 3 input is selected 011_1111b - sai1_tx_sync_out[1] input is selected 100_0000b - sai1_rx_sync_out[2] input is selected All other values are reserved.

1. sai_tx_sync_out is Transmit Frame Sync for multi-SAI synchronous operation.
2. sai_rx_sync_out is Receive Frame Sync for multi-SAI synchronous operation.

26.5.1.4 Trigger Register for CTIMER (TIMER0TRIG - TIMER2TRIG)

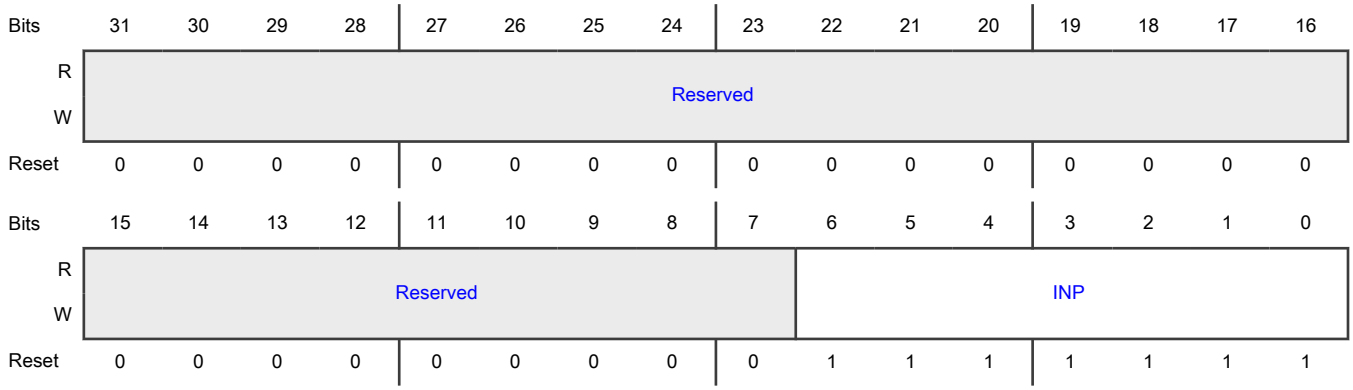
Offset

Register	Offset
TIMER0TRIG	30h
TIMER1TRIG	50h
TIMER2TRIG	70h

Function

This is the trigger register for CTIMER.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 INP	Input number for CTIMER 000_0000b - CT_INP0 input is selected 000_0001b - CT_INP1 input is selected 000_0010b - CT_INP2 input is selected 000_0011b - CT_INP3 input is selected 000_0100b - CT_INP4 input is selected 000_0101b - CT_INP5 input is selected 000_0110b - CT_INP6 input is selected 000_0111b - CT_INP7 input is selected 000_1000b - CT_INP8 input is selected 000_1001b - CT_INP9 input is selected 000_1010b - CT_INP10 input is selected 000_1011b - CT_INP11 input is selected 000_1100b - CT_INP12 input is selected 000_1101b - CT_INP13 input is selected 000_1110b - CT_INP14 input is selected 000_1111b - CT_INP15 input is selected 001_0000b - CT_INP16 input is selected 001_0001b - CT_INP17 input is selected 001_0010b - CT_INP18 input is selected 001_0011b - CT_INP19 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001_0100b - usb0 start of frame input is selected 001_0101b - usb1 start of frame input is selected 001_0110b - DCDC_BURST_ACTIVE input is selected 001_0111b - sai0_tx_sync_out[1] input is selected 001_1000b - sai0_rx_sync_out[2] input is selected 001_1001b - ADC0_IRQ input is selected 001_1010b - ADC1_IRQ input is selected 001_1011b - CMP0_OUT input is selected 001_1100b - CMP1_OUT input is selected 001_1101b - CMP2_OUT input is selected 001_1110b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 001_1111b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 010_0000b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 010_0001b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 010_0010b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 010_0011b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 010_0100b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 010_0101b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 010_0110b - QDC0_CMP/POS_MATCH input is selected 010_0111b - QDC1_CMP/POS_MATCH input is selected 010_1000b - EVTG_OUT0A input is selected 010_1001b - EVTG_OUT0B input is selected 010_1010b - EVTG_OUT1A input is selected 010_1011b - EVTG_OUT1B input is selected 010_1100b - EVTG_OUT2A input is selected 010_1101b - EVTG_OUT2B input is selected 010_1110b - EVTG_OUT3A input is selected 010_1111b - EVTG_OUT3B input is selected 011_0000b - Reserved 011_0001b - Reserved 011_0010b - LP_FLEXCOMM0 trig 0 input is selected 011_0011b - LP_FLEXCOMM0 trig 1 input is selected 011_0100b - LP_FLEXCOMM0 trig 2 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011_0101b - LP_FLEXCOMM1 trig 0 input is selected 011_0110b - LP_FLEXCOMM1 trig 1 input is selected 011_0111b - LP_FLEXCOMM1 trig 2 input is selected 011_1000b - LP_FLEXCOMM2 trig 0 input is selected 011_1001b - LP_FLEXCOMM2 trig 1 input is selected 011_1010b - LP_FLEXCOMM2 trig 2 input is selected 011_1011b - LP_FLEXCOMM3 trig 0 input is selected 011_1100b - LP_FLEXCOMM3 trig 1 input is selected 011_1101b - LP_FLEXCOMM3 trig 2 input is selected 011_1110b - LP_FLEXCOMM3 trig 3 input is selected 011_1111b - sai1_tx_sync_out[1] input is selected 100_0000b - sai1_rx_sync_out[2] input is selected All other values are reserved.

1. sai_tx_sync_out is Transmit Frame Sync for multi-SAI synchronous operation.
2. sai_rx_sync_out is Receive Frame Sync for multi-SAI synchronous operation.

26.5.1.5 Inputmux Register for SMARTDMA Arch B Inputs (SMARTDMAARCHB_INMUX0 - SMARTDMAARCHB_INMUX7)

Offset

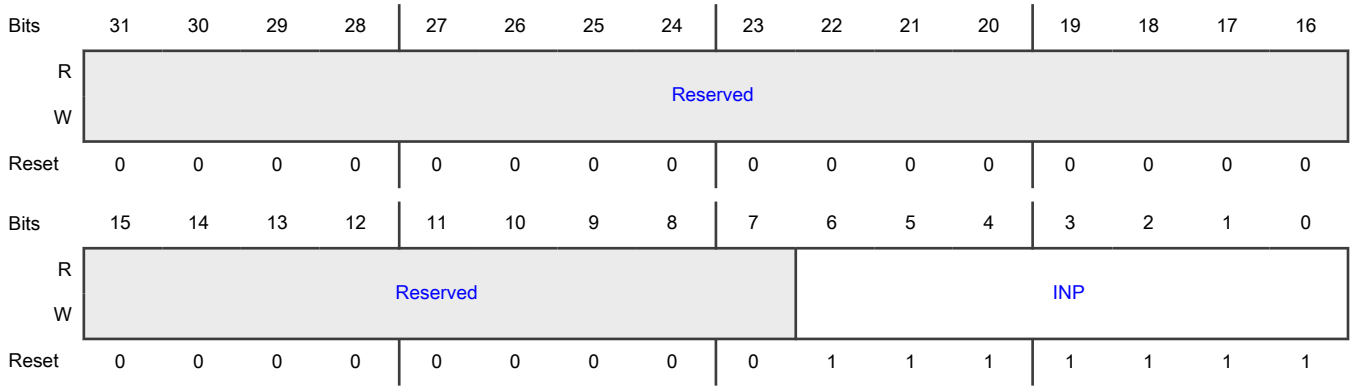
For a = 0 to 7:

Register	Offset
SMARTDMAARCHB_INMUXa	A0h + (a × 4h)

Function

This register is used to select SMARTDMA ArchB inputs.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 INP	Input number select to SmartDMA ARCHB input 000_0000b - FlexIO interrupt is selected as input 000_0001b - GPIO P0_1 input is selected 000_0010b - GPIO P0_2 input is selected 000_0011b - GPIO P0_3 input is selected 000_0100b - GPIO P0_4 input is selected 000_0101b - GPIO P0_5 input is selected 000_0110b - GPIO P0_6 input is selected 000_0111b - GPIO P0_7 input is selected 000_1000b - GPIO P0_8 input is selected 000_1001b - GPIO P0_9 input is selected 000_1010b - GPIO P0_10 input is selected 000_1011b - GPIO P0_11 input is selected 000_1100b - GPIO P0_12 input is selected 000_1101b - GPIO P0_13 input is selected 000_1110b - GPIO P0_14 input is selected 000_1111b - GPIO P0_15 input is selected 001_0000b - SCT0 SCT_OUT8 input is selected 001_0001b - SCT0 SCT_OUT9 input is selected 001_0010b - Reserved 001_0011b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001_0100b - MRT0 MRT_CH0_IRQ input is selected 001_0101b - MRT0 MRT_CH1_IRQ input is selected 001_0110b - CTIMER4_MAT3 input is selected 001_0111b - CTIMER4_MAT2 input is selected 001_1000b - CTIMER3_MAT3 input is selected 001_1001b - CTIMER3_MAT2 input is selected 001_1010b - CTIMER1_MAT3 input is selected 001_1011b - CTIMER1_MAT2 input is selected 001_1100b - UTICK0 UTICK_IRQ input is selected 001_1101b - WWDT0 WDT0_IRQ input is selected 001_1110b - ADC0 ADC0_IRQ input is selected 001_1111b - CMP0_IRQ input is selected 010_0000b - Reserved 010_0001b - LP_FLEXCOMM7_IRQ input is selected 010_0010b - LP_FLEXCOMM6_IRQ input is selected 010_0011b - LP_FLEXCOMM5_IRQ input is selected 010_0100b - LP_FLEXCOMM4_IRQ input is selected 010_0101b - LP_FLEXCOMM3_IRQ input is selected 010_0110b - LP_FLEXCOMM2_IRQ input is selected 010_0111b - LP_FLEXCOMM1_IRQ input is selected 010_1000b - LP_FLEXCOMM0_IRQ input is selected 010_1001b - DMA0_IRQ input is selected 010_1010b - DMA1_IRQ input is selected 010_1011b - SYS_IRQ[1] input is selected 010_1100b - RTC_COMBO_IRQ input is selected 010_1101b - ARM_TXEV input is selected 010_1110b - PINT0 GPIO_INT_BMATCH input is selected 010_1111b - Reserved 011_0000b - Reserved 011_0001b - CMP0_OUT input is selected 011_0010b - usb0 start of frame input is selected 011_0011b - usb1 start of frame input is selected 011_0100b - OSTIMER0 OS_EVENT_TIMER_IRQ input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011_0101b - ADC1_IRQ input is selected 011_0110b - CMP0_IRQ/CMP1_IRQ/CMP2_IRQ input is selected 011_0111b - DAC0_IRQ input is selected 011_1000b - DAC1_IRQ/DAC2_IRQ input is selected 011_1001b - PWM0_IRQ input is selected 011_1010b - PWM1_IRQ input is selected 011_1011b - QDC0_IRQ input is selected 011_1100b - QDC1_IRQ input is selected 011_1101b - EVTG_OUT0A input is selected 011_1110b - EVTG_OUT1A input is selected 011_1111b - Reserved 100_0000b - Reserved 100_0001b - GPIO1_alias0 GPIO1 Pin Event Trig 0 input is selected 100_0010b - GPIO1_alias1 GPIO1 Pin Event Trig 1 input is selected 100_0011b - GPIO2_alias0 GPIO2 Pin Event Trig 0 input is selected 100_0100b - GPIO2_alias1 GPIO2 Pin Event Trig 1 input is selected 100_0101b - GPIO3_alias0 GPIO3 Pin Event Trig 0 input is selected 100_0110b - GPIO3_alias1 GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

1. SYS_IRQ combines the CDOG IRQ, WWDT IRQ, MBC secure violation IRQ, Secure AHB Matrix secure violation IRQ, GDET IRQ, ELS S50 error IRQ, PKC error IRQ, and VBAT IRQ using the logical OR operation.

26.5.1.6 Pin Interrupt Select (PINTSEL0 - PINTSEL7)

Offset

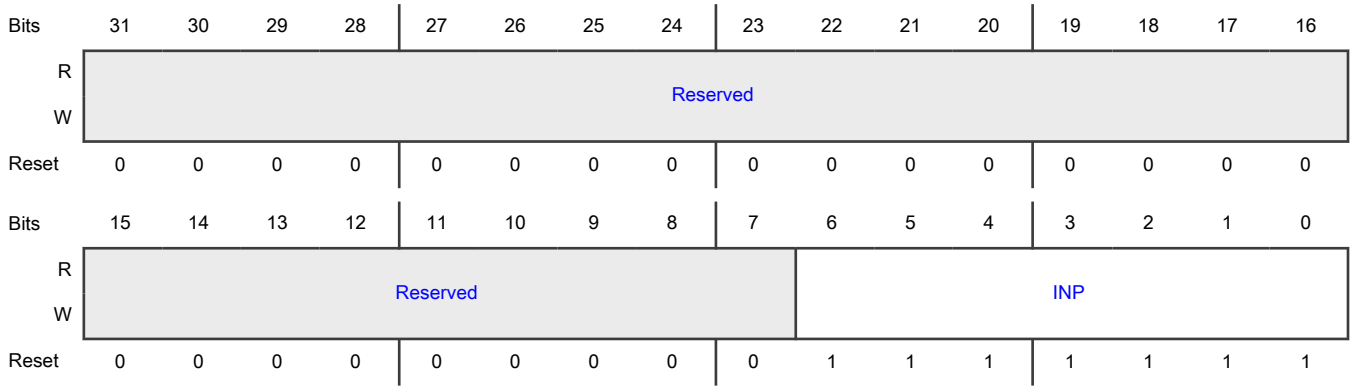
For a = 0 to 7:

Register	Offset
PINTSELa	C0h + (a × 4h)

Function

Each of these eight registers selects one pin from among ports 0 and 1 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 8 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 31 for pins P0_0 to P0_31 to the INP bits. Port 1 pins correspond to pin numbers 32 to 63. For example, setting INP to 0x5 in PINTSEL0 selects pin P0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 INP	Pin number select for pin interrupt or pattern match engine input. For PIOx_y: INP = (x * 32) + y. PIO0_0 to PIO1_31 correspond to numbers 0 to 63. 000_0000b - GPIO P0_0 input is selected 000_0001b - GPIO P0_1 input is selected 000_0010b - GPIO P0_2 input is selected 000_0011b - GPIO P0_3 input is selected 000_0100b - GPIO P0_4 input is selected 000_0101b - GPIO P0_5 input is selected 000_0110b - GPIO P0_6 input is selected 000_0111b - GPIO P0_7 input is selected 000_1000b - GPIO P0_8 input is selected 000_1001b - GPIO P0_9 input is selected 000_1010b - GPIO P0_10 input is selected 000_1011b - GPIO P0_11 input is selected 000_1100b - GPIO P0_12 input is selected 000_1101b - GPIO P0_13 input is selected 000_1110b - GPIO P0_14 input is selected 000_1111b - GPIO P0_15 input is selected 001_0000b - GPIO P0_16 input is selected 001_0001b - GPIO P0_17 input is selected 001_0010b - GPIO P0_18 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001_0011b - GPIO P0_19 input is selected 001_0100b - GPIO P0_20 input is selected 001_0101b - GPIO P0_21 input is selected 001_0110b - GPIO P0_22 input is selected 001_0111b - GPIO P0_23 input is selected 001_1000b - GPIO P0_24 input is selected 001_1001b - GPIO P0_25 input is selected 001_1010b - GPIO P0_26 input is selected 001_1011b - GPIO P0_27 input is selected 001_1100b - GPIO P0_28 input is selected 001_1101b - GPIO P0_29 input is selected 001_1110b - GPIO P0_30 input is selected 001_1111b - GPIO P0_31 input is selected 010_0000b - GPIO P1_0 input is selected 010_0001b - GPIO P1_1 input is selected 010_0010b - GPIO P1_2 input is selected 010_0011b - GPIO P1_3 input is selected 010_0100b - GPIO P1_4 input is selected 010_0101b - GPIO P1_5 input is selected 010_0110b - GPIO P1_6 input is selected 010_0111b - GPIO P1_7 input is selected 010_1000b - GPIO P1_8 input is selected 010_1001b - GPIO P1_9 input is selected 010_1010b - GPIO P1_10 input is selected 010_1011b - GPIO P1_11 input is selected 010_1100b - GPIO P1_12 input is selected 010_1101b - GPIO P1_13 input is selected 010_1110b - GPIO P1_14 input is selected 010_1111b - GPIO P1_15 input is selected 011_0000b - GPIO P1_16 input is selected 011_0001b - GPIO P1_17 input is selected 011_0010b - GPIO P1_18 input is selected 011_0011b - GPIO P1_19 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011_0100b - GPIO P1_20 input is selected 011_0101b - GPIO P1_21 input is selected 011_0110b - GPIO P1_22 input is selected 011_0111b - GPIO P1_23 input is selected 011_1000b - Reserved 011_1001b - Reserved 011_1010b - Reserved 011_1011b - Reserved 011_1100b - Reserved 011_1101b - Reserved 011_1110b - GPIO P1_30 input is selected 011_1111b - GPIO P1_31 input is selected All other values are reserved.

26.5.1.7 Selection for Frequency Measurement Reference Clock (FREQMEAS_REF)

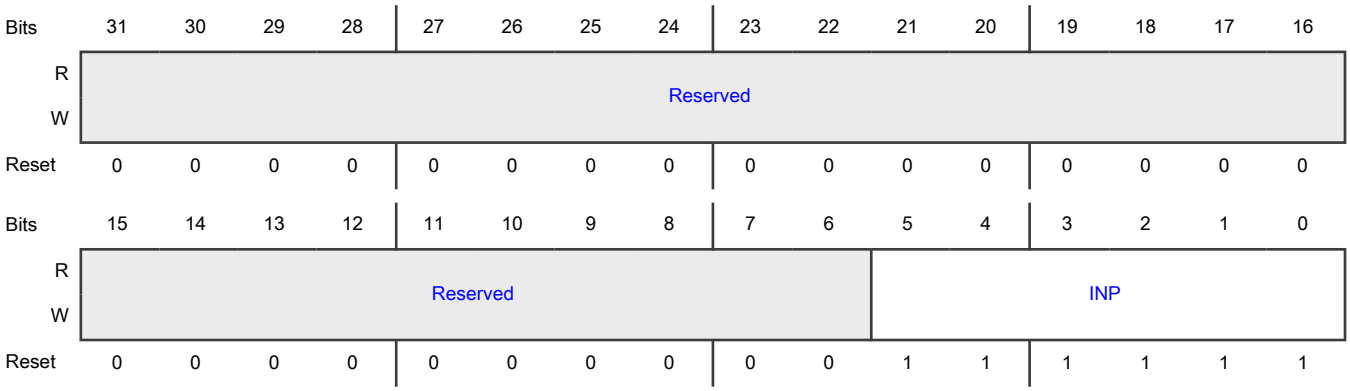
Offset

Register	Offset
FREQMEAS_REF	180h

Function

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	Clock source number (binary value) for frequency measure function reference clock. 00_0000b - clk_in (output of clk_in or XTAL mux in Clockgen) input is selected 00_0001b - FRO_12M input is selected 00_0010b - FRO_144M input is selected 00_0011b - Reserved 00_0100b - OSC_32K input is selected 00_0101b - CPU/system_clk input is selected 00_0110b - FREQME_CLK_IN0 input is selected 00_0111b - FREQME_CLK_IN1 input is selected 00_1000b - EVTG_OUT0A input is selected 00_1001b - EVTG_OUT1A input is selected All other values are reserved.

26.5.1.8 Selection for Frequency Measurement Target Clock (FREQMEAS_TAR)

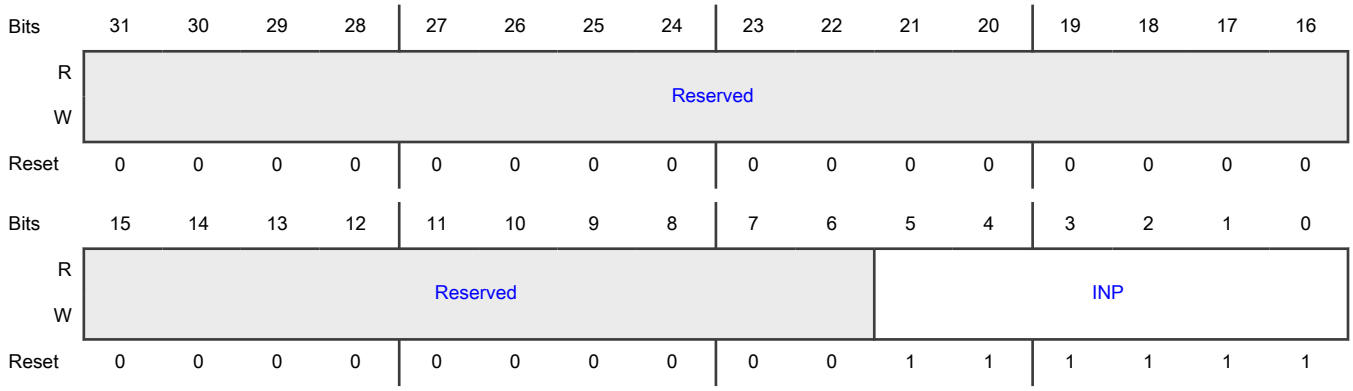
Offset

Register	Offset
FREQMEAS_TAR	184h

Function

This register selects a clock for the target clock of the frequency measure function. By default, no clock is selected.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	Clock source number (binary value) for frequency measure function target clock. 00_0000b - clk_in (output of clk_in or XTAL mux in Clockgen) input is selected 00_0001b - FRO_12M input is selected 00_0010b - FRO_144M input is selected 00_0011b - Reserved 00_0100b - OSC_32K input is selected 00_0101b - CPU/system_clk input is selected 00_0110b - FREQME_CLK_IN0 input is selected 00_0111b - FREQME_CLK_IN1 input is selected 00_1000b - EVTG_OUT0A input is selected 00_1001b - EVTG_OUT1A input is selected All other values are reserved.

26.5.1.9 Capture Select Register for CTIMER Inputs (CTIMER3CAP0 - CTIMER4CAP3)

Offset

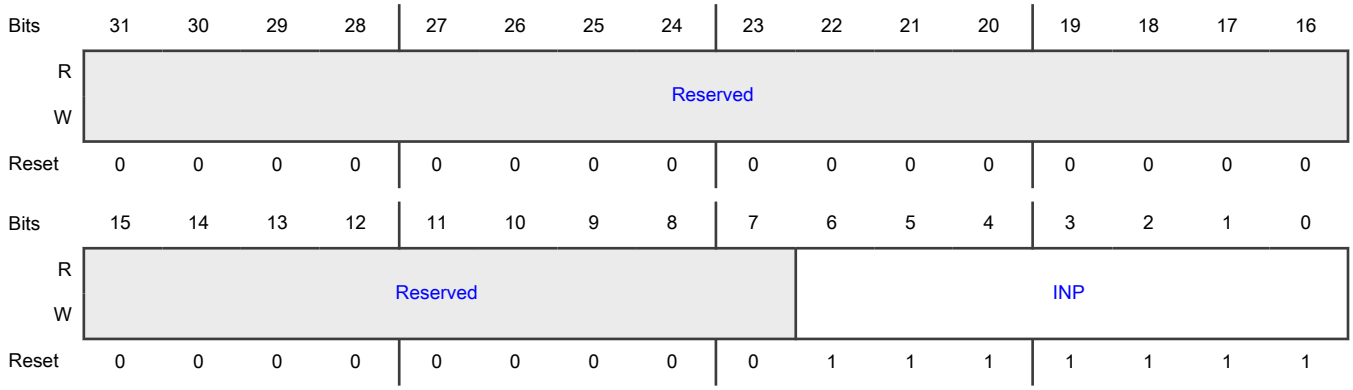
For a = 3 to 4; b = 0 to 3:

Register	Offset
CTIMERaCAPb	140h + (a × 20h) + (b × 4h)

Function

For each of the 5 standard timers, numbered i = 0 to 4 there are 4 CTIMERiCAPTj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources. The output of TIMER0CAPT0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPT1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPT3 Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 INP	Input number for CTIMER 000_0000b - CT_INP0 input is selected 000_0001b - CT_INP1 input is selected 000_0010b - CT_INP2 input is selected 000_0011b - CT_INP3 input is selected 000_0100b - CT_INP4 input is selected 000_0101b - CT_INP5 input is selected 000_0110b - CT_INP6 input is selected 000_0111b - CT_INP7 input is selected 000_1000b - CT_INP8 input is selected 000_1001b - CT_INP9 input is selected 000_1010b - CT_INP10 input is selected 000_1011b - CT_INP11 input is selected 000_1100b - CT_INP12 input is selected 000_1101b - CT_INP13 input is selected 000_1110b - CT_INP14 input is selected 000_1111b - CT_INP15 input is selected 001_0000b - CT_INP16 input is selected 001_0001b - CT_INP17 input is selected 001_0010b - CT_INP18 input is selected 001_0011b - CT_INP19 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001_0100b - usb0 start of frame input is selected 001_0101b - usb1 start of frame input is selected 001_0110b - DCDC_BURST_ACTIVE input is selected 001_0111b - sai0_tx_sync_out[1] input is selected 001_1000b - sai0_rx_sync_out[2] input is selected 001_1001b - ADC0 ADC0_IRQ input is selected 001_1010b - ADC0 ADC1_IRQ input is selected 001_1011b - CMP0_OUT input is selected 001_1100b - CMP1_OUT input is selected 001_1101b - CMP2_OUT input is selected 001_1110b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 001_1111b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 010_0000b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 010_0001b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 010_0010b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 010_0011b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 010_0100b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 010_0101b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 010_0110b - QDC0_CMP/POS_MATCH input is selected 010_0111b - QDC1_CMP/POS_MATCH input is selected 010_1000b - EVTG_OUT0A input is selected 010_1001b - EVTG_OUT0B input is selected 010_1010b - EVTG_OUT1A input is selected 010_1011b - EVTG_OUT1B input is selected 010_1100b - EVTG_OUT2A input is selected 010_1101b - EVTG_OUT2B input is selected 010_1110b - EVTG_OUT3A input is selected 010_1111b - EVTG_OUT3B input is selected 011_0000b - Reserved 011_0001b - Reserved 011_0010b - LP_FLEXCOMM0 trig 0 input is selected 011_0011b - LP_FLEXCOMM0 trig 1 input is selected 011_0100b - LP_FLEXCOMM0 trig 2 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011_0101b - LP_FLEXCOMM1 trig 0 input is selected 011_0110b - LP_FLEXCOMM1 trig 1 input is selected 011_0111b - LP_FLEXCOMM1 trig 2 input is selected 011_1000b - LP_FLEXCOMM2 trig 0 input is selected 011_1001b - LP_FLEXCOMM2 trig 1 input is selected 011_1010b - LP_FLEXCOMM2 trig 2 input is selected 011_1011b - LP_FLEXCOMM3 trig 0 input is selected 011_1100b - LP_FLEXCOMM3 trig 1 input is selected 011_1101b - LP_FLEXCOMM3 trig 2 input is selected 011_1110b - LP_FLEXCOMM3 trig 3 input is selected 011_1111b - sai1_tx_sync_out[1] input is selected 100_0000b - sai1_rx_sync_out[2] input is selected All other values are reserved.

1. sai_tx_sync_out is Transmit Frame Sync for multi-SAI synchronous operation.
2. sai_rx_sync_out is Receive Frame Sync for multi-SAI synchronous operation.

26.5.1.10 Trigger Register for CTIMER (TIMER3TRIG - TIMER4TRIG)

Offset

Register	Offset
TIMER3TRIG	1B0h
TIMER4TRIG	1D0h

Function

This is the trigger register for CTIMER.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								INP							
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Fields

Field	Function
31-7 —	Reserved
6-0 INP	Input number for CTIMER 000_0000b - CT_INP0 input is selected 000_0001b - CT_INP1 input is selected 000_0010b - CT_INP2 input is selected 000_0011b - CT_INP3 input is selected 000_0100b - CT_INP4 input is selected 000_0101b - CT_INP5 input is selected 000_0110b - CT_INP6 input is selected 000_0111b - CT_INP7 input is selected 000_1000b - CT_INP8 input is selected 000_1001b - CT_INP9 input is selected 000_1010b - CT_INP10 input is selected 000_1011b - CT_INP11 input is selected 000_1100b - CT_INP12 input is selected 000_1101b - CT_INP13 input is selected 000_1110b - CT_INP14 input is selected 000_1111b - CT_INP15 input is selected 001_0000b - CT_INP16 input is selected 001_0001b - CT_INP17 input is selected 001_0010b - CT_INP18 input is selected 001_0011b - CT_INP19 input is selected 001_0100b - usb0 start of frame input is selected 001_0101b - usb1 start of frame input is selected 001_0110b - DCDC_BURST_ACTIVE input is selected 001_0111b - sai0_tx_sync_out[1] input is selected 001_1000b - sai0_rx_sync_out[2] input is selected 001_1001b - ADC0 ADC0_IRQ input is selected 001_1010b - ADC0 ADC1_IRQ input is selected 001_1011b - CMP0_OUT input is selected 001_1100b - CMP1_OUT input is selected 001_1101b - CMP2_OUT input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001_1110b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 001_1111b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 010_0000b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 010_0001b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 010_0010b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 010_0011b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 010_0100b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 010_0101b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 010_0110b - QDC0_CMP/POS_MATCH input is selected 010_0111b - QDC1_CMP/POS_MATCH input is selected 010_1000b - EVTG_OUT0A input is selected 010_1001b - EVTG_OUT0B input is selected 010_1010b - EVTG_OUT1A input is selected 010_1011b - EVTG_OUT1B input is selected 010_1100b - EVTG_OUT2A input is selected 010_1101b - EVTG_OUT2B input is selected 010_1110b - EVTG_OUT3A input is selected 010_1111b - EVTG_OUT3B input is selected 011_0000b - Reserved 011_0001b - Reserved 011_0010b - LP_FLEXCOMM0 trig 0 input is selected 011_0011b - LP_FLEXCOMM0 trig 1 input is selected 011_0100b - LP_FLEXCOMM0 trig 2 input is selected 011_0101b - LP_FLEXCOMM1 trig 0 input is selected 011_0110b - LP_FLEXCOMM1 trig 1 input is selected 011_0111b - LP_FLEXCOMM1 trig 2 input is selected 011_1000b - LP_FLEXCOMM2 trig 0 input is selected 011_1001b - LP_FLEXCOMM2 trig 1 input is selected 011_1010b - LP_FLEXCOMM2 trig 2 input is selected 011_1011b - LP_FLEXCOMM3 trig 0 input is selected 011_1100b - LP_FLEXCOMM3 trig 1 input is selected 011_1101b - LP_FLEXCOMM3 trig 2 input is selected 011_1110b - LP_FLEXCOMM3 trig 3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011_1111b - sai1_tx_sync_out[1] input is selected 100_0000b - sai1_rx_sync_out[2] input is selected All other values are reserved.

1. sai_tx_sync_out is Transmit Frame Sync for multi-SAI synchronous operation.
2. sai_rx_sync_out is Receive Frame Sync for multi-SAI synchronous operation.

26.5.1.11 CMP0 Input Connections (CMP0_TRIG)

Offset

Register	Offset
CMP0_TRIG	260h

Function

This register selects the CMP0 SAMPLE/WINDOW input.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	CMP0 input trigger 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT6 input is selected 00_0010b - SCT_OUT4 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT6 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER0_MAT0 input is selected 00_1001b - CTIMER4_MAT0 input is selected 00_1010b - Reserved 00_1011b - Reserved 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC1_tcomp[0] input is selected 00_1111b - Reserved 01_0000b - Reserved 01_0001b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 01_0010b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 01_0011b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 01_0100b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 01_0101b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 01_0110b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 01_0111b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 01_1000b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 01_1001b - QDC0_CMP/POS_MATCH input is selected 01_1010b - QDC1_CMP/POS_MATCH input is selected 01_1011b - EVTG_OUT0A input is selected 01_1100b - EVTG_OUT0B input is selected 01_1101b - EVTG_OUT1A input is selected 01_1110b - EVTG_OUT1B input is selected 01_1111b - EVTG_OUT2A input is selected 10_0000b - EVTG_OUT2B input is selected 10_0001b - EVTG_OUT3A input is selected 10_0010b - EVTG_OUT3B input is selected 10_0011b - LPTMR0 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_0100b - LPTMR1 input is selected 10_0101b - GPIO2 Pin Event Trig 0 input is selected 10_0110b - GPIO2 Pin Event Trig 1 input is selected 10_0111b - GPIO3 Pin Event Trig 0 input is selected 10_1000b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.12 ADC Trigger Input Connections (ADC0_TRIG0 - ADC0_TRIG3)

Offset

Register	Offset
ADC0_TRIG0	280h
ADC0_TRIG1	284h
ADC0_TRIG2	288h
ADC0_TRIG3	28Ch

Function
This register selects the ADC0 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								TRIGIN							
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Fields

Field	Function
31-8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 TRIGIN	ADC0 trigger inputs 0000_0000b - PINT PIN_INT0 input is selected 0000_0001b - PINT PIN_INT1 input is selected 0000_0010b - SCT0 SCT_OUT4 input is selected 0000_0011b - SCT0 SCT_OUT5 input is selected 0000_0100b - SCT0 SCT_OUT9 input is selected 0000_0101b - CTIMER0_MAT3 input is selected 0000_0110b - CTIMER1_MAT3 input is selected 0000_0111b - CTIMER2_MAT3 input is selected 0000_1000b - CTIMER3_MAT3 input is selected 0000_1001b - CTIMER4_MAT3 input is selected 0000_1010b - DCDC_Burst_Done_Trig input is selected 0000_1011b - Reserved 0000_1100b - PINT GPIO_INT_BMAT input is selected 0000_1101b - ADC0_tcomp[0] input is selected 0000_1110b - ADC0_tcomp[1] input is selected 0000_1111b - ADC0_tcomp[2] input is selected 0001_0000b - ADC0_tcomp[3] input is selected 0001_0001b - ADC1_tcomp[0] input is selected 0001_0010b - ADC1_tcomp[1] input is selected 0001_0011b - ADC1_tcomp[2] input is selected 0001_0100b - ADC1_tcomp[3] input is selected 0001_0101b - CMP0_OUT input is selected 0001_0110b - CMP1_OUT input is selected 0001_0111b - CMP2_OUT input is selected 0001_1000b - PWM0_SM0_MUX_TRIG0 input is selected 0001_1001b - PWM0_SM0_MUX_TRIG1 input is selected 0001_1010b - PWM0_SM1_MUX_TRIG0 input is selected 0001_1011b - PWM0_SM1_MUX_TRIG1 input is selected 0001_1100b - PWM0_SM2_MUX_TRIG0 input is selected 0001_1101b - PWM0_SM2_MUX_TRIG1 input is selected 0001_1110b - PWM0_SM3_MUX_TRIG0 input is selected 0001_1111b - PWM0_SM3_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0010_0000b - PWM1_SM0_MUX_TRIG0 input is selected 0010_0001b - PWM1_SM0_MUX_TRIG1 input is selected 0010_0010b - PWM1_SM1_MUX_TRIG0 input is selected 0010_0011b - PWM1_SM1_MUX_TRIG1 input is selected 0010_0100b - PWM1_SM2_MUX_TRIG0 input is selected 0010_0101b - PWM1_SM2_MUX_TRIG1 input is selected 0010_0110b - PWM1_SM3_MUX_TRIG0 input is selected 0010_0111b - PWM1_SM3_MUX_TRIG1 input is selected 0010_1000b - QDC0_CMP/POS_MATCH input is selected 0010_1001b - QDC1_CMP/POS_MATCH input is selected 0010_1010b - EVTG_OUT0A input is selected 0010_1011b - EVTG_OUT0B input is selected 0010_1100b - EVTG_OUT1A input is selected 0010_1101b - EVTG_OUT1B input is selected 0010_1110b - EVTG_OUT2A input is selected 0010_1111b - EVTG_OUT2B input is selected 0011_0000b - EVTG_OUT3A input is selected 0011_0001b - EVTG_OUT3B input is selected 0011_0010b - LPTMR0 input is selected 0011_0011b - LPTMR1 input is selected 0011_0100b - FlexIO CH0 input is selected 0011_0101b - FlexIO CH1 input is selected 0011_0110b - FlexIO CH2 input is selected 0011_0111b - FlexIO CH3 input is selected 0011_1000b - SINC Filter CH0 Conversion Complete input is selected 0011_1001b - SINC Filter CH1 Conversion Complete input is selected 0011_1010b - SINC Filter CH2 Conversion Complete input is selected 0011_1011b - SINC Filter CH3 Conversion Complete input is selected 0011_1100b - SINC Filter CH4 Conversion Complete input is selected 0011_1101b - GPIO2 Pin Event Trig 0 input is selected 0011_1110b - GPIO2 Pin Event Trig 1 input is selected 0011_1111b - GPIO3 Pin Event Trig 0 input is selected 0100_0000b - GPIO3 Pin Event Trig 1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	All other values are reserved.

26.5.1.13 ADC Trigger Input Connections (ADC1_TRIG0 - ADC1_TRIG3)

Offset

Register	Offset
ADC1_TRIG0	2C0h
ADC1_TRIG1	2C4h
ADC1_TRIG2	2C8h
ADC1_TRIG3	2CCh

Function
This register selects the ADC1 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								TRIGIN							
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Fields

Field	Function
31-8 —	Reserved
7-0 TRIGIN	ADC1 trigger inputs 0000_0000b - PINT PIN_INT0 input is selected 0000_0001b - PINT PIN_INT2 input is selected 0000_0010b - SCT0 SCT_OUT4 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000_0011b - SCT0 SCT_OUT5 input is selected 0000_0100b - SCT0 SCT_OUT3 input is selected 0000_0101b - CTIMER0_MAT3 input is selected 0000_0110b - CTIMER1_MAT3 input is selected 0000_0111b - CTIMER2_MAT3 input is selected 0000_1000b - CTIMER3_MAT2 input is selected 0000_1001b - CTIMER4_MAT1 input is selected 0000_1010b - DCDC_Burst_Done_Trig input is selected 0000_1011b - Reserved 0000_1100b - PINT GPIO_INT_BMAT input is selected 0000_1101b - ADC0_tcomp[0] input is selected 0000_1110b - ADC0_tcomp[1] input is selected 0000_1111b - ADC0_tcomp[2] input is selected 0001_0000b - ADC0_tcomp[3] input is selected 0001_0001b - ADC1_tcomp[0] input is selected 0001_0010b - ADC1_tcomp[1] input is selected 0001_0011b - ADC1_tcomp[2] input is selected 0001_0100b - ADC1_tcomp[3] input is selected 0001_0101b - CMP0_OUT input is selected 0001_0110b - CMP1_OUT input is selected 0001_0111b - CMP2_OUT input is selected 0001_1000b - PWM0_SM0_MUX_TRIG0 input is selected 0001_1001b - PWM0_SM0_MUX_TRIG1 input is selected 0001_1010b - PWM0_SM1_MUX_TRIG0 input is selected 0001_1011b - PWM0_SM1_MUX_TRIG1 input is selected 0001_1100b - PWM0_SM2_MUX_TRIG0 input is selected 0001_1101b - PWM0_SM2_MUX_TRIG1 input is selected 0001_1110b - PWM0_SM3_MUX_TRIG0 input is selected 0001_1111b - PWM0_SM3_MUX_TRIG1 input is selected 0010_0000b - PWM1_SM0_MUX_TRIG0 input is selected 0010_0001b - PWM1_SM0_MUX_TRIG1 input is selected 0010_0010b - PWM1_SM1_MUX_TRIG0 input is selected 0010_0011b - PWM1_SM1_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0010_0100b - PWM1_SM2_MUX_TRIG0 input is selected 0010_0101b - PWM1_SM2_MUX_TRIG1 input is selected 0010_0110b - PWM1_SM3_MUX_TRIG0 input is selected 0010_0111b - PWM1_SM3_MUX_TRIG1 input is selected 0010_1000b - QDC0_CMP/POS_MATCH input is selected 0010_1001b - QDC1_CMP/POS_MATCH input is selected 0010_1010b - EVTG_OUT0A input is selected 0010_1011b - EVTG_OUT0B input is selected 0010_1100b - EVTG_OUT1A input is selected 0010_1101b - EVTG_OUT1B input is selected 0010_1110b - EVTG_OUT2A input is selected 0010_1111b - EVTG_OUT2B input is selected 0011_0000b - EVTG_OUT3A input is selected 0011_0001b - EVTG_OUT3B input is selected 0011_0010b - LPTMR0 input is selected 0011_0011b - LPTMR1 input is selected 0011_0100b - FlexIO CH0 input is selected 0011_0101b - FlexIO CH1 input is selected 0011_0110b - FlexIO CH2 input is selected 0011_0111b - FlexIO CH3 input is selected 0011_1000b - SINC Filter CH0 Conversion Complete input is selected 0011_1001b - SINC Filter CH1 Conversion Complete input is selected 0011_1010b - SINC Filter CH2 Conversion Complete input is selected 0011_1011b - SINC Filter CH3 Conversion Complete input is selected 0011_1100b - SINC Filter CH4 Conversion Complete input is selected 0011_1101b - GPIO2 Pin Event Trig 0 input is selected 0011_1110b - GPIO2 Pin Event Trig 1 input is selected 0011_1111b - GPIO3 Pin Event Trig 0 input is selected 0100_0000b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.14 DAC0 Trigger Inputs (DAC0_TRIG)

Offset

Register	Offset
DAC0_TRIG	300h

Function

This register selects the DAC0 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	DAC0 trigger input 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT3 input is selected 00_0010b - SCT0 SCT_OUT4 input is selected 00_0011b - SCT0 SCT_OUT5 input is selected 00_0100b - SCT0 SCT_OUT0 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_1011b - Reserved 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC1_tcomp[0] input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - LPTMR0 input is selected 01_1011b - LPTMR1 input is selected 01_1100b - GPIO2 Pin Event Trig 0 input is selected 01_1101b - GPIO2 Pin Event Trig 1 input is selected 01_1110b - GPIO3 Pin Event Trig 0 input is selected 01_1111b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

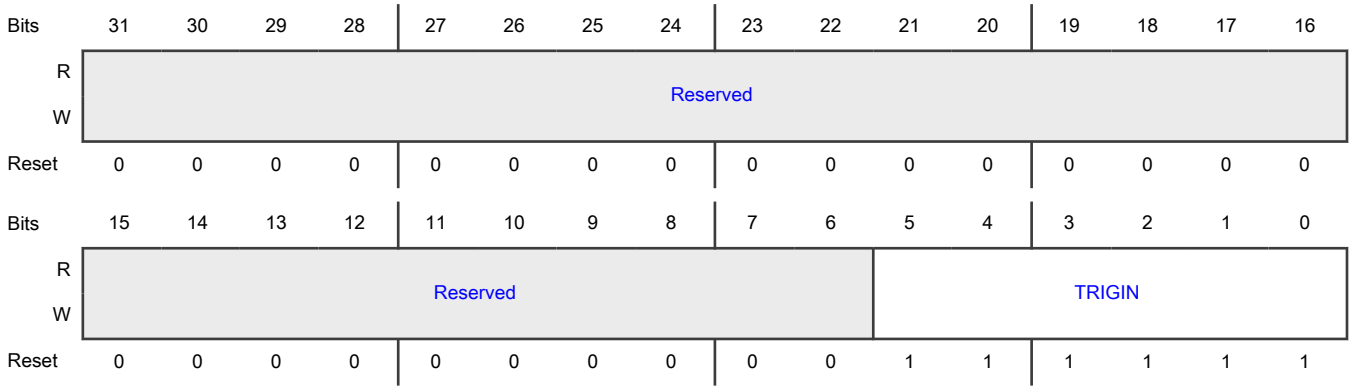
26.5.1.15 DAC1 Trigger Inputs (DAC1_TRIG)

Offset

Register	Offset
DAC1_TRIG	320h

Function
This register selects the DAC1 trigger inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	DAC1 trigger input 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT0 SCT_OUT4 input is selected 00_0011b - SCT0 SCT_OUT5 input is selected 00_0100b - SCT0 SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT1 input is selected 00_1001b - CTIMER3_MAT1 input is selected 00_1010b - Reserved 00_1011b - Reserved 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[1] input is selected 00_1110b - ADC1_tcomp[1] input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - LPTMR0 input is selected 01_1011b - LPTMR1 input is selected 01_1100b - GPIO2 Pin Event Trig 0 input is selected 01_1101b - GPIO2 Pin Event Trig 1 input is selected 01_1110b - GPIO3 Pin Event Trig 0 input is selected 01_1111b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.16 DAC2 Trigger Inputs (DAC2_TRIG)

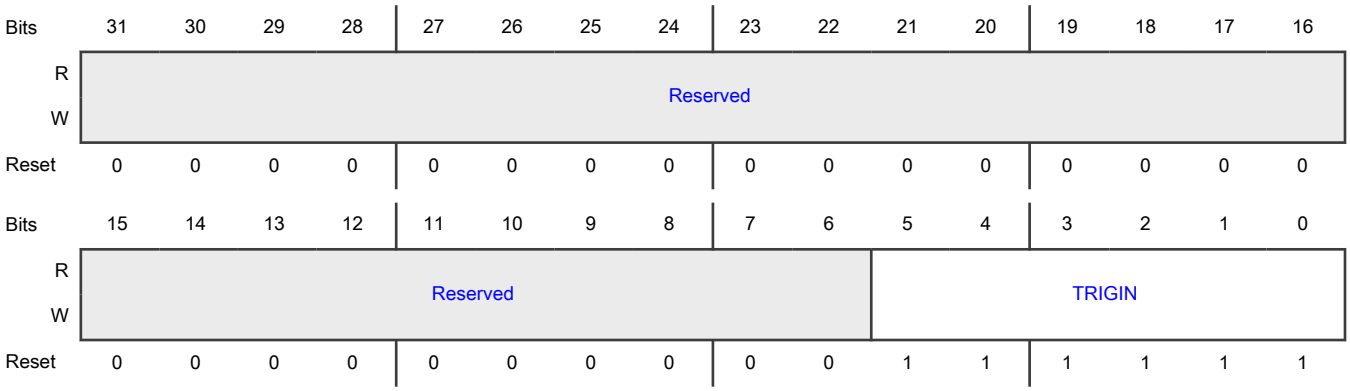
Offset

Register	Offset
DAC2_TRIG	340h

Function

This register selects the DAC2 trigger inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	DAC2 trigger input 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - SCT0 SCT_OUT4 input is selected 00_0011b - SCT0 SCT_OUT5 input is selected 00_0100b - SCT0 SCT_OUT2 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT2 input is selected 00_1001b - CTIMER3_MAT2 input is selected 00_1010b - Reserved 00_1011b - Reserved 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[2] input is selected 00_1110b - ADC1_tcomp[2] input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - LPTMR0 input is selected 01_1011b - LPTMR1 input is selected 01_1100b - GPIO2 Pin Event Trig 0 input is selected 01_1101b - GPIO2 Pin Event Trig 1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - GPIO3 Pin Event Trig 0 input is selected 01_1111b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.17 QDC0 Trigger Input Connections (QDC0_TRIG)

Offset

Register	Offset
QDC0_TRIG	360h

Function

This register selects the QDC0 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC0 trigger input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

26.5.1.18 QDC0 Input Connections (QDC0_HOME)

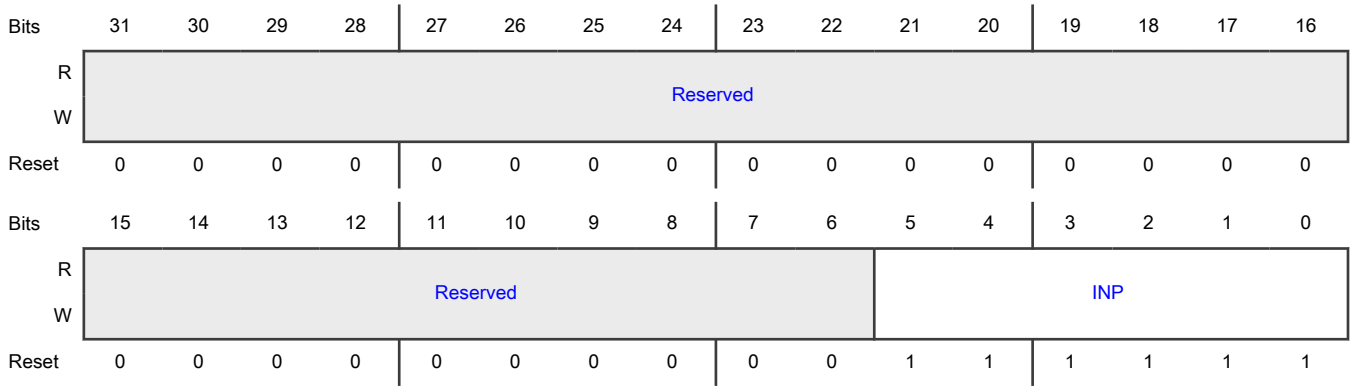
Offset

Register	Offset
QDC0_HOME	364h

Function

This register selects the QDC0 HOME inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC0 HOME input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT0 SCT_OUT4 input is selected 00_0011b - SCT0 SCT_OUT5 input is selected 00_0100b - SCT0 SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

26.5.1.19 QDC0 Input Connections (QDC0_INDEX)

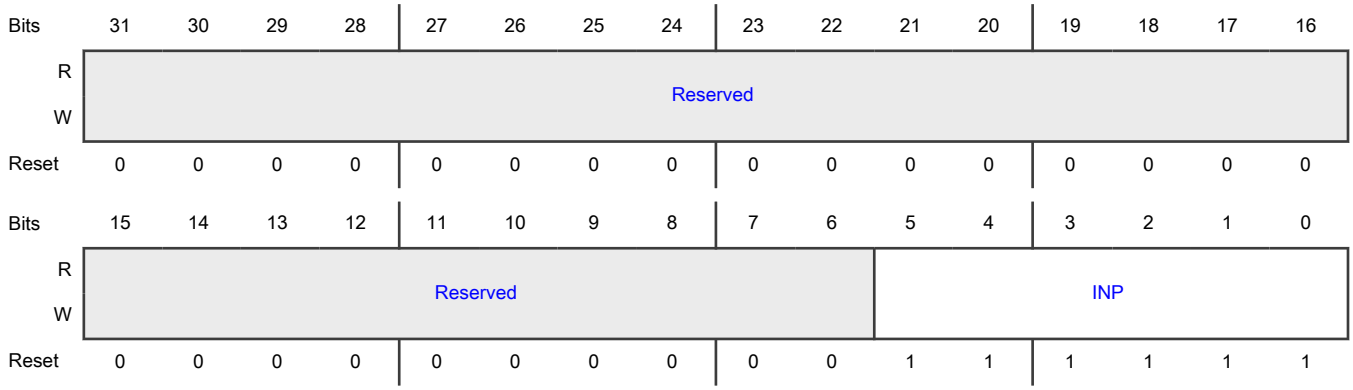
Offset

Register	Offset
QDC0_INDEX	368h

Function

This register selects the QDC0 INDEX inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC0 INDEX input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

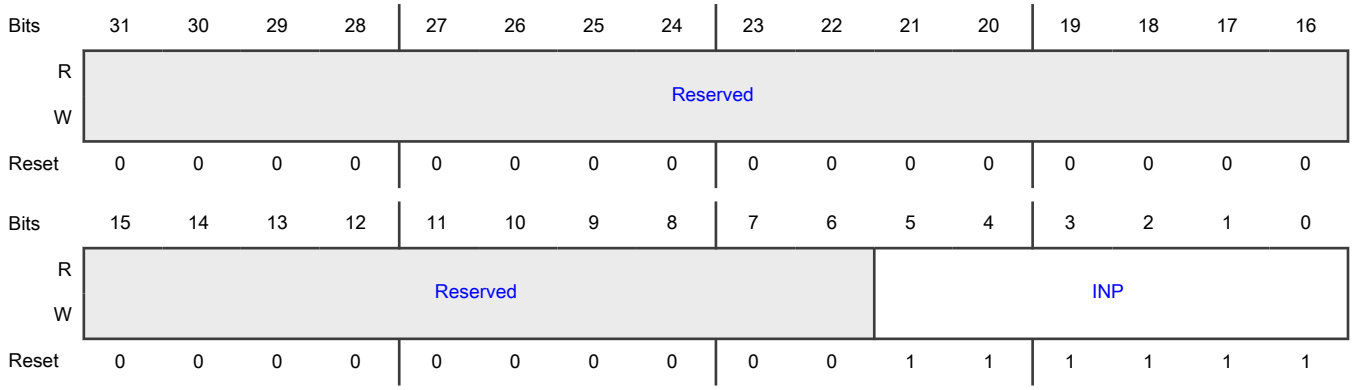
26.5.1.20 QDC0 Input Connections (QDC0_PHASEB)

Offset

Register	Offset
QDC0_PHASEB	36Ch

Function
This register selects the QDC0 PHASEB inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC0 PHASEB input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

26.5.1.21 QDC0 Input Connections (QDC0_PHASEA)

Offset

Register	Offset
QDC0_PHASEA	370h

Function

This register selects the QDC0 PHASEA inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-0 INP	QDC0 PHASEA input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

26.5.1.22 QDC1 Trigger Input Connections (QDC1_TRIG)

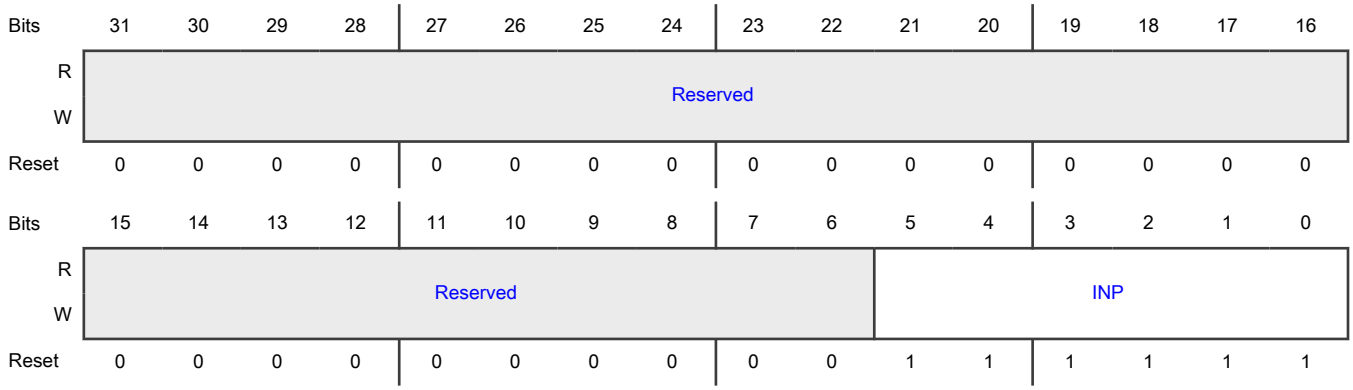
Offset

Register	Offset
QDC1_TRIG	380h

Function

This register selects the QDC1 trigger inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC1 trigger input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

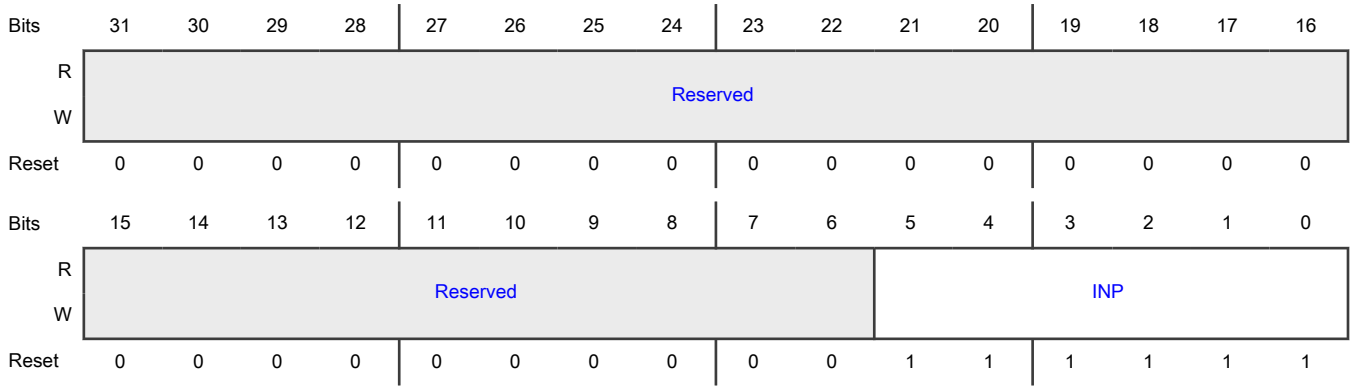
26.5.1.23 QDC1 Input Connections (QDC1_HOME)

Offset

Register	Offset
QDC1_HOME	384h

Function
This register selects the QDC1 HOME inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC1 HOME input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT0 SCT_OUT4 input is selected 00_0011b - SCT0 SCT_OUT5 input is selected 00_0100b - SCT0 SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

26.5.1.24 QDC1 Input Connections (QDC1_INDEX)

Offset

Register	Offset
QDC1_INDEX	388h

Function

This register selects the QDC1 INDEX inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-0 INP	QDC1 INDEX input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

26.5.1.25 QDC1 Input Connections (QDC1_PHASEB)

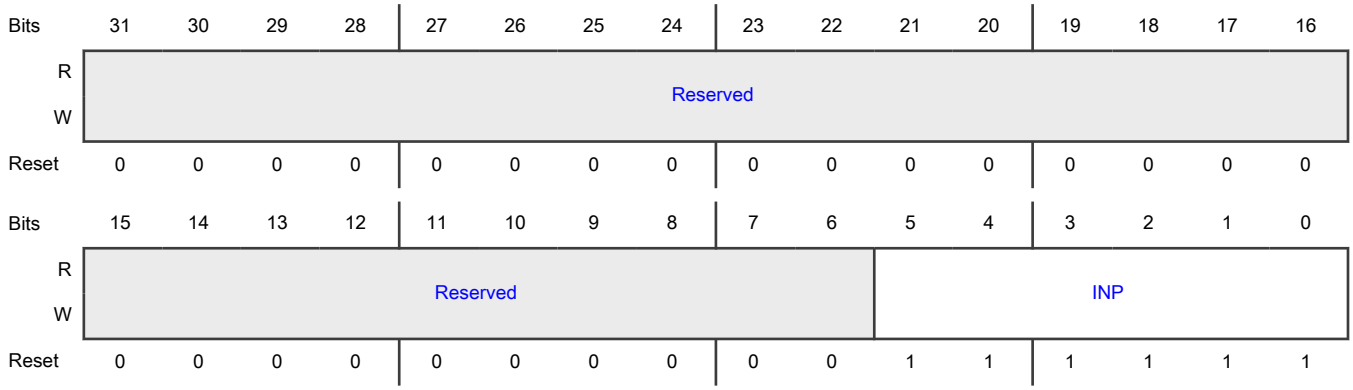
Offset

Register	Offset
QDC1_PHASEB	38Ch

Function

This register selects the QDC1 PHASEB inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC1 PHASEB input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

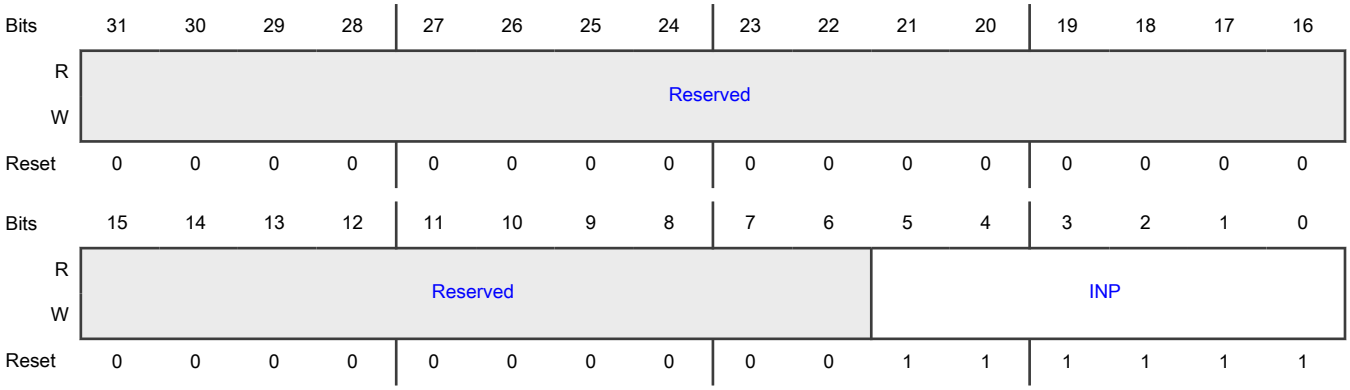
26.5.1.26 QDC1 Input Connections (QDC1_PHASEA)

Offset

Register	Offset
QDC1_PHASEA	390h

Function
This register selects the QDC1 PHASEA inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	QDC1 PHASEA input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER1_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected All other values are reserved.

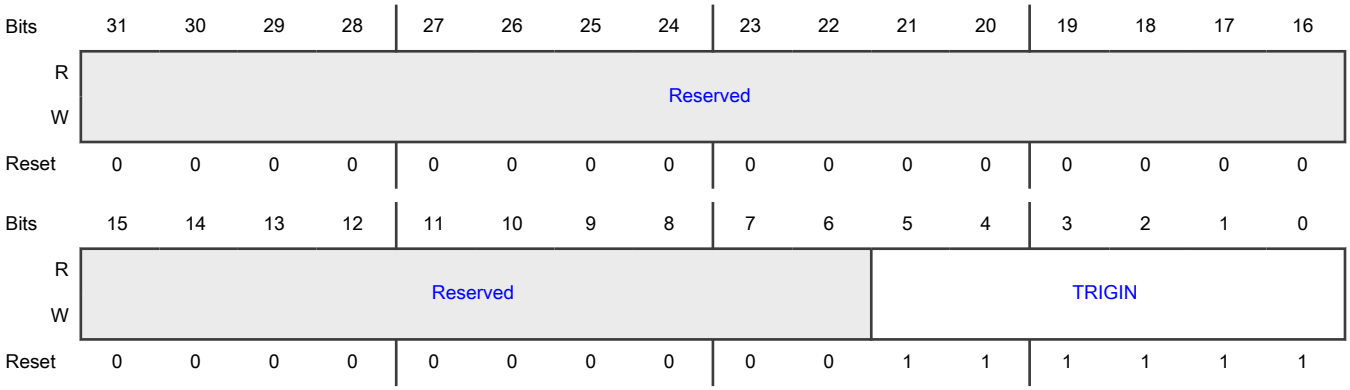
26.5.1.27 PWM0 External Synchronization (FlexPWM0_SM0_EXTSYNC - FlexPWM0_SM3_EXTSYNC)

Offset

Register	Offset
FlexPWM0_SM0_EXTSYNC	3A0h
FlexPWM0_SM1_EXTSYNC	3A4h
FlexPWM0_SM2_EXTSYNC	3A8h
FlexPWM0_SM3_EXTSYNC	3ACh

Function
This register selects the PWM0 EXTSYNC inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	EXTSYNC input connections for PWM0 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT2 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER4_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected 11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.28 PWM0 Input Trigger Connections (FlexPWM0_SM0_EXTA - FlexPWM0_SM3_EXTA)

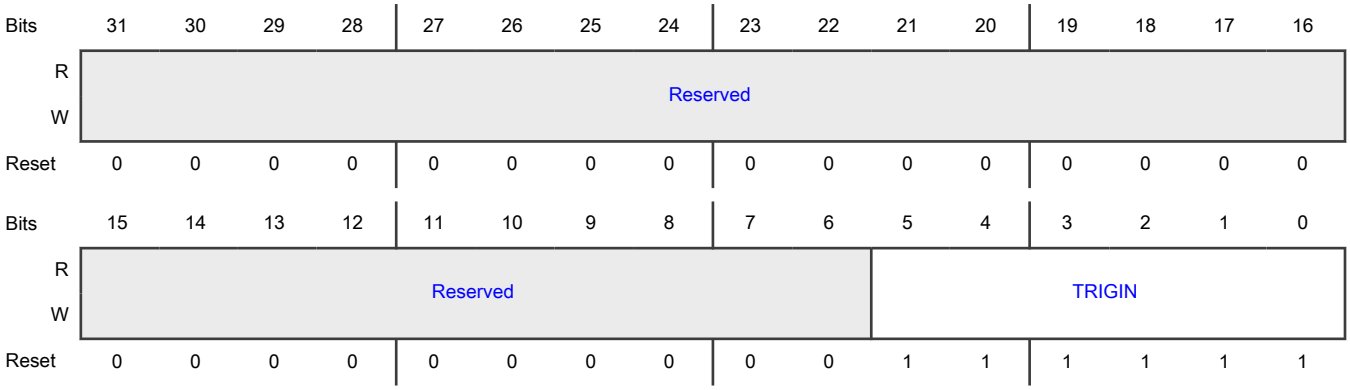
Offset

Register	Offset
FlexPWM0_SM0_EXTA	3B0h
FlexPWM0_SM1_EXTA	3B4h
FlexPWM0_SM2_EXTA	3B8h
FlexPWM0_SM3_EXTA	3BCh

Function

This register selects the PWM0 EXTA inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	EXTA input connections for PWM0 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT2 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER4_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected 11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

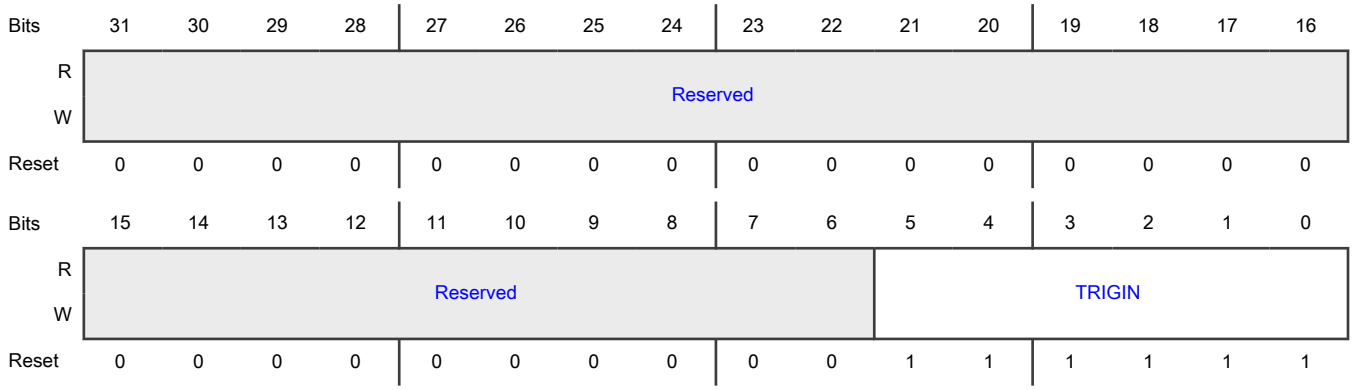
26.5.1.29 PWM0 External Force Trigger Connections (FlexPWM0_EXTFORCE)

Offset

Register	Offset
FlexPWM0_EXTFORCE	3C0h

Function
This register selects the PWM0 EXTFORCE inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	EXTFORCE input connections for PWM0 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT2 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER4_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected 01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.30 PWM0 Fault Input Trigger Connections (FlexPWM0_FAULT0 - FlexPWM0_FAULT3)

Offset

Register	Offset
FlexPWM0_FAULT0	3C4h
FlexPWM0_FAULT1	3C8h
FlexPWM0_FAULT2	3CCh
FlexPWM0_FAULT3	3D0h

Function
This register selects the PWM0 FAULT inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	FAULT input connections for PWM0 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT2 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER4_MAT0 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM1_SM0_MUX_TRIG0 input is selected 01_1001b - PWM1_SM0_MUX_TRIG1 input is selected 01_1010b - PWM1_SM1_MUX_TRIG0 input is selected 01_1011b - PWM1_SM1_MUX_TRIG1 input is selected 01_1100b - PWM1_SM2_MUX_TRIG0 input is selected 01_1101b - PWM1_SM2_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - PWM1_SM3_MUX_TRIG0 input is selected 01_1111b - PWM1_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected 11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.31 PWM1 External Synchronization (FlexPWM1_SM0_EXTSYNC - FlexPWM1_SM3_EXTSYNC)

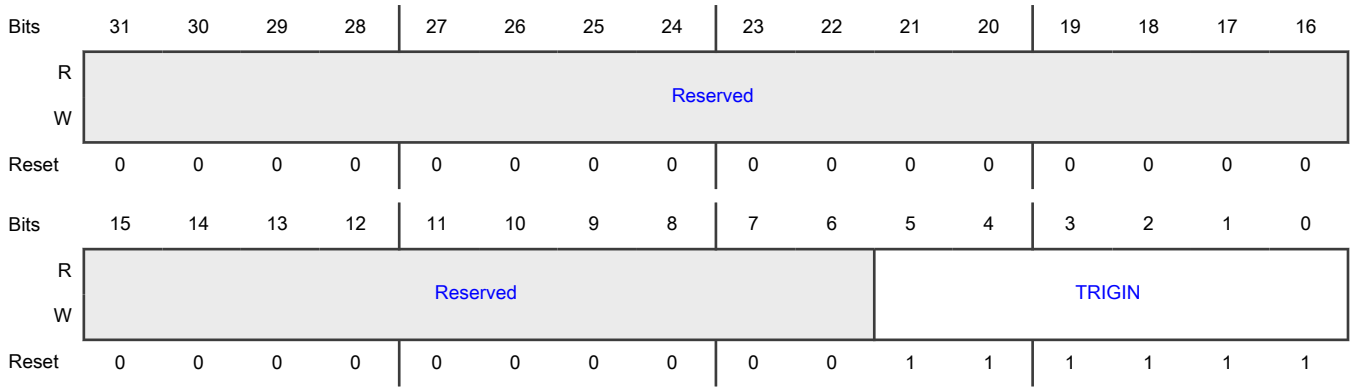
Offset

Register	Offset
FlexPWM1_SM0_EXTSYNC	3E0h
FlexPWM1_SM1_EXTSYNC	3E4h
FlexPWM1_SM2_EXTSYNC	3E8h
FlexPWM1_SM3_EXTSYNC	3ECh

Function

This register selects the PWM1 EXTSYNC inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	EXTSYNC input connections for PWM1 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT2 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT1 input is selected 00_1001b - CTIMER4_MAT1 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM0_SM0_MUX_TRIG0 input is selected 01_1001b - PWM0_SM0_MUX_TRIG1 input is selected 01_1010b - PWM0_SM1_MUX_TRIG0 input is selected 01_1011b - PWM0_SM1_MUX_TRIG1 input is selected 01_1100b - PWM0_SM2_MUX_TRIG0 input is selected 01_1101b - PWM0_SM2_MUX_TRIG1 input is selected 01_1110b - PWM0_SM3_MUX_TRIG0 input is selected 01_1111b - PWM0_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected 11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.32 PWM1 Input EXTA Connections (FlexPWM1_SM0_EXTA - FlexPWM1_SM3_EXTA)

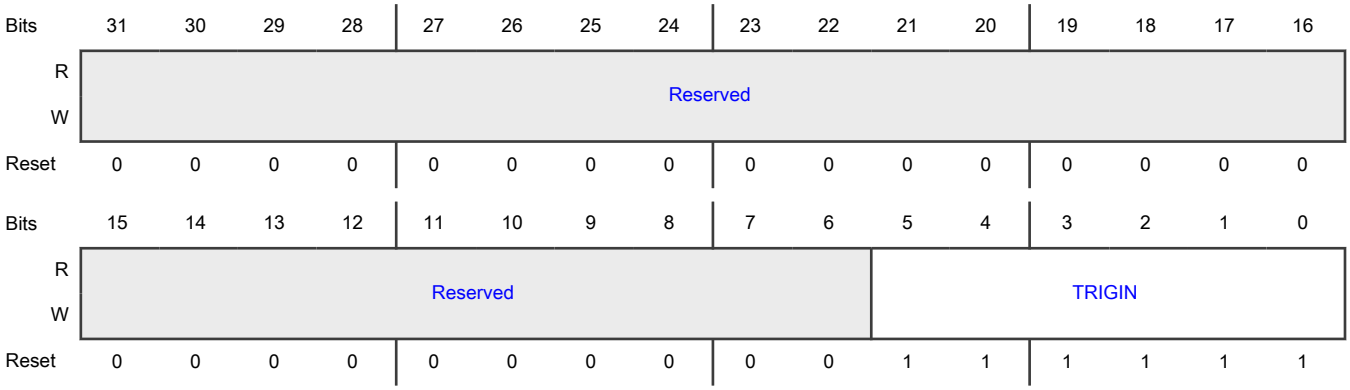
Offset

Register	Offset
FlexPWM1_SM0_EXTA	3F0h
FlexPWM1_SM1_EXTA	3F4h
FlexPWM1_SM2_EXTA	3F8h
FlexPWM1_SM3_EXTA	3FCh

Function

This register selects the PWM1 EXTA inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	EXTA input connections for PWM1 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT2 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT3 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT1 input is selected 00_1001b - CTIMER4_MAT1 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM0_SM0_MUX_TRIG0 input is selected 01_1001b - PWM0_SM0_MUX_TRIG1 input is selected 01_1010b - PWM0_SM1_MUX_TRIG0 input is selected 01_1011b - PWM0_SM1_MUX_TRIG1 input is selected 01_1100b - PWM0_SM2_MUX_TRIG0 input is selected 01_1101b - PWM0_SM2_MUX_TRIG1 input is selected 01_1110b - PWM0_SM3_MUX_TRIG0 input is selected 01_1111b - PWM0_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected 11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.33 PWM1 External Force Trigger Connections (FlexPWM1_EXTFORCE)

Offset

Register	Offset
FlexPWM1_EXTFORCE	400h

Function

This register selects the PWM1 EXTFORCE inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	EXTFORCE input connections for PWM1 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT2 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT3 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER2_MAT1 input is selected 00_1001b - CTIMER4_MAT1 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM0_SM0_MUX_TRIG0 input is selected 01_1001b - PWM0_SM0_MUX_TRIG1 input is selected 01_1010b - PWM0_SM1_MUX_TRIG0 input is selected 01_1011b - PWM0_SM1_MUX_TRIG1 input is selected 01_1100b - PWM0_SM2_MUX_TRIG0 input is selected 01_1101b - PWM0_SM2_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - PWM0_SM3_MUX_TRIG0 input is selected 01_1111b - PWM0_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected 10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected 11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.34 PWM1 Fault Input Trigger Connections (FlexPWM1_FAULT0 - FlexPWM1_FAULT3)

Offset

Register	Offset
FlexPWM1_FAULT0	404h
FlexPWM1_FAULT1	408h
FlexPWM1_FAULT2	40Ch
FlexPWM1_FAULT3	410h

Function

This register selects the PWM1 FAULT inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										TRIGIN					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	FAULT input connections for PWM1 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT2 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT3 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_1000b - CTIMER2_MAT1 input is selected 00_1001b - CTIMER4_MAT1 input is selected 00_1010b - Reserved 00_1011b - ARM_TXEV input is selected 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM0_SM0_MUX_TRIG0 input is selected 01_1001b - PWM0_SM0_MUX_TRIG1 input is selected 01_1010b - PWM0_SM1_MUX_TRIG0 input is selected 01_1011b - PWM0_SM1_MUX_TRIG1 input is selected 01_1100b - PWM0_SM2_MUX_TRIG0 input is selected 01_1101b - PWM0_SM2_MUX_TRIG1 input is selected 01_1110b - PWM0_SM3_MUX_TRIG0 input is selected 01_1111b - PWM0_SM3_MUX_TRIG1 input is selected 10_0000b - QDC0_CMP/POS_MATCH input is selected 10_0001b - QDC1_CMP/POS_MATCH input is selected 10_0010b - EVTG_OUT0A input is selected 10_0011b - EVTG_OUT0B input is selected 10_0100b - EVTG_OUT1A input is selected 10_0101b - EVTG_OUT1B input is selected 10_0110b - EVTG_OUT2A input is selected 10_0111b - EVTG_OUT2B input is selected 10_1000b - EVTG_OUT3A input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_1001b - EVTG_OUT3B input is selected 10_1010b - TRIG_IN0 input is selected 10_1011b - TRIG_IN1 input is selected 10_1100b - TRIG_IN2 input is selected 10_1101b - TRIG_IN3 input is selected 10_1110b - TRIG_IN4 input is selected 10_1111b - TRIG_IN5 input is selected 11_0000b - TRIG_IN6 input is selected 11_0001b - TRIG_IN7 input is selected 11_0010b - TRIG_IN8 input is selected 11_0011b - TRIG_IN9 input is selected 11_0100b - SINC Filter CH0 sync Break input is selected 11_0101b - SINC Filter CH1 sync Break input is selected 11_0110b - SINC Filter CH2 sync Break input is selected 11_0111b - SINC Filter CH3 sync Break input is selected 11_1000b - SINC Filter CH4 sync Break input is selected 11_1001b - GPIO2 Pin Event Trig 0 input is selected 11_1010b - GPIO2 Pin Event Trig 1 input is selected 11_1011b - GPIO3 Pin Event Trig 0 input is selected 11_1100b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.35 PWM0 External Clock Trigger (PWM0_EXT_CLK)

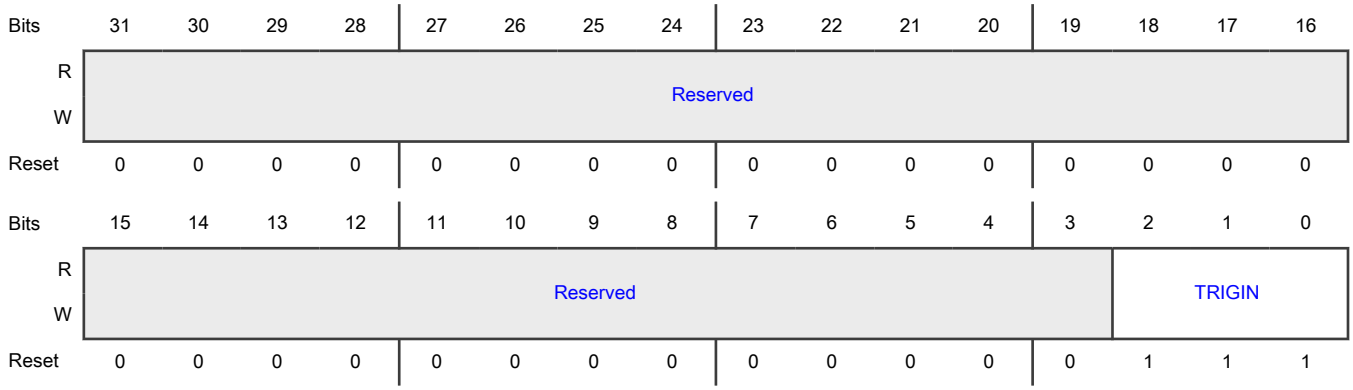
Offset

Register	Offset
PWM0_EXT_CLK	420h

Function

PWM0 external clock trigger connections

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 TRIGIN	EXT_CLK input connections for PWM0 000b - FRO16K input is selected 001b - OSC_32k input is selected 010b - EVTG_OUT0A input is selected 011b - EVTG_OUT1A input is selected 100b - TRIG_IN0 input is selected 101b - TRIG_IN7 input is selected All other values are reserved.

26.5.1.36 PWM1 External Clock Trigger (PWM1_EXT_CLK)

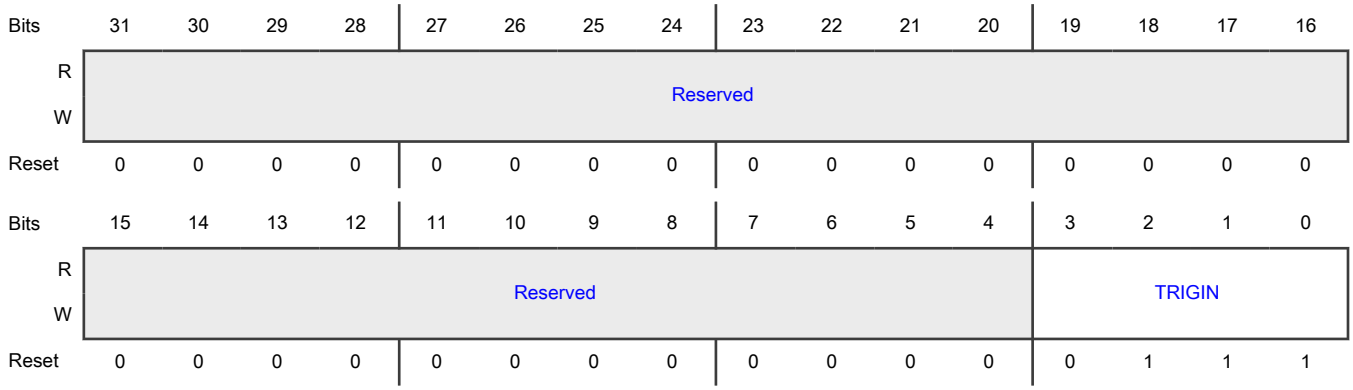
Offset

Register	Offset
PWM1_EXT_CLK	424h

Function

PWM1 external clock trigger connections

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 TRIGIN	EXT_CLK input connections for PWM1 0000b - FRO16K input is selected 0001b - OSC_32k input is selected 0010b - EVTG_OUT0A input is selected 0011b - EVTG_OUT1A input is selected 0100b - TRIG_IN0 input is selected 0101b - TRIG_IN7 input is selected All other values are reserved.

26.5.1.37 EVTG Trigger Input Connections (EVTG_TRIG0 - EVTG_TRIG15)

Offset

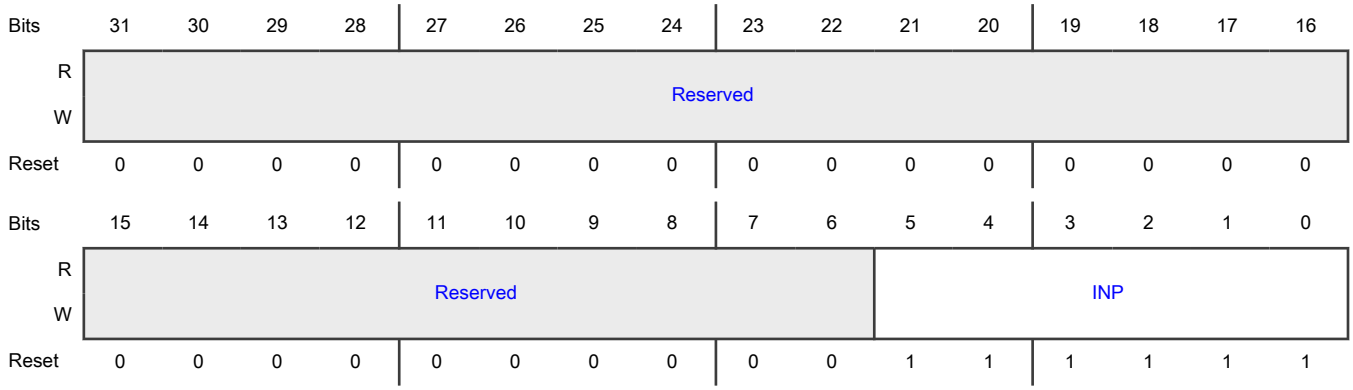
For a = 0 to 15:

Register	Offset
EVTG_TRIGa	440h + (a × 4h)

Function

This register is used to select the EVTG trigger inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	EVTG trigger input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT1 input is selected 00_0010b - SCT_OUT0 input is selected 00_0011b - SCT_OUT1 input is selected 00_0100b - SCT_OUT2 input is selected 00_0101b - SCT_OUT3 input is selected 00_0110b - CTIMER0_MAT3 input is selected 00_0111b - CTIMER1_MAT3 input is selected 00_1000b - CTIMER2_MAT3 input is selected 00_1001b - CTIMER2_MAT2 input is selected 00_1010b - CTIMER3_MAT2 input is selected 00_1011b - CTIMER4_MAT2 input is selected 00_1100b - Reserved 00_1101b - PINT GPIO_INT_BMAT input is selected 00_1110b - ADC0_IRQ input is selected 00_1111b - ADC1_IRQ input is selected 01_0000b - ADC0_tcomp[0] input is selected 01_0001b - ADC0_tcomp[1] input is selected 01_0010b - ADC0_tcomp[2] input is selected 01_0011b - ADC0_tcomp[3] input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0100b - ADC1_tcomp[0] input is selected 01_0101b - ADC1_tcomp[1] input is selected 01_0110b - ADC1_tcomp[2] input is selected 01_0111b - ADC1_tcomp[3] input is selected 01_1000b - CMP0_OUT input is selected 01_1001b - CMP1_OUT input is selected 01_1010b - CMP2_OUT input is selected 01_1011b - PWM0_SM0_MUX_TRIG0 input is selected 01_1100b - PWM0_SM0_MUX_TRIG1 input is selected 01_1101b - PWM0_SM1_MUX_TRIG0 input is selected 01_1110b - PWM0_SM1_MUX_TRIG1 input is selected 01_1111b - PWM0_SM2_MUX_TRIG0 input is selected 10_0000b - PWM0_SM2_MUX_TRIG1 input is selected 10_0001b - PWM0_SM3_MUX_TRIG0 input is selected 10_0010b - PWM0_SM3_MUX_TRIG1 input is selected 10_0011b - PWM1_SM0_MUX_TRIG0 input is selected 10_0100b - PWM1_SM0_MUX_TRIG1 input is selected 10_0101b - PWM1_SM1_MUX_TRIG0 input is selected 10_0110b - PWM1_SM1_MUX_TRIG1 input is selected 10_0111b - PWM1_SM2_MUX_TRIG0 input is selected 10_1000b - PWM1_SM2_MUX_TRIG1 input is selected 10_1001b - PWM1_SM3_MUX_TRIG0 input is selected 10_1010b - PWM1_SM3_MUX_TRIG1 input is selected 10_1011b - QDC0_CMP/POS_MATCH input is selected 10_1100b - QDC1_CMP/POS_MATCH input is selected 10_1101b - TRIG_IN0 input is selected 10_1110b - TRIG_IN1 input is selected 10_1111b - TRIG_IN2 input is selected 11_0000b - TRIG_IN3 input is selected 11_0001b - LPTMR0 input is selected 11_0010b - LPTMR1 input is selected 11_0011b - SINC Filter CH0 Break input is selected 11_0100b - SINC Filter CH1 Break input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11_0101b - SINC Filter CH2 Break input is selected 11_0110b - SINC Filter CH3 Break input is selected 11_0111b - SINC Filter CH4 Break input is selected 11_1000b - Reserved 11_1001b - Reserved All other values are reserved.

26.5.1.38 USB-FS Trigger Input Connections (USBFS_TRIG)

Offset

Register	Offset
USBFS_TRIG	480h

Function

This register is used to select the USB-FS trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved												INP			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Fields

Field	Function
31-4 —	Reserved
3-0 INP	USB-FS trigger input connections. The trigger output of LP_FLEXCOMM is an input of peripheral INPUTMUX. 0000b - LP_FLEXCOMM 0 trigger out [3] input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0001b - LP_FLEXCOMM 1 trigger out [3] input is selected 0010b - LP_FLEXCOMM 2 trigger out [3] input is selected 0011b - LP_FLEXCOMM 3 trigger out [3] input is selected 0100b - LP_FLEXCOMM 4 trigger out [3] input is selected 0101b - LP_FLEXCOMM 5 trigger out [3] input is selected 0110b - LP_FLEXCOMM 6 trigger out [3] input is selected 0111b - LP_FLEXCOMM 7 trigger out [3] input is selected 1000b - LP_FLEXCOMM 8 trigger out [3] input is selected 1001b - LP_FLEXCOMM 9 trigger out [3] input is selected All other values are reserved.

26.5.1.39 TSI Trigger Input Connections (TSI_TRIG)

Offset

Register	Offset
TSI_TRIG	4A0h

Function

This register is used to select the TSI trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														INP	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Fields

Field	Function
31-2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1-0 INP	TSI trigger input connections 00b - LPTMR0 input is selected 01b - LPTMR1 input is selected All other values are reserved.

26.5.1.40 EXT Trigger Connections (EXT_TRIG0 - EXT_TRIG7)

Offset

For a = 0 to 7:

Register	Offset
EXT_TRIGa	4C0h + (a × 4h)

Function

The EXT_TRIGa regsiters select the source of TRIG_OUTa pins.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	TRIG_OUTa pin input connections 00_0000b - PINT PIN_INT0 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_0001b - PINT PIN_INT1 input is selected 00_0010b - ADC0_IRQ input is selected 00_0011b - ADC1_IRQ input is selected 00_0100b - ADC0_tcomp[0] input is selected 00_0101b - ADC1_tcomp[0] input is selected 00_0110b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 00_0111b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 00_1000b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 00_1001b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 00_1010b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 00_1011b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 00_1100b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 00_1101b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 00_1110b - QDC0_CMP/POS_MATCH input is selected 00_1111b - QDC1_CMP/POS_MATCH input is selected 01_0000b - EVTG_OUT0A input is selected 01_0001b - EVTG_OUT0B input is selected 01_0010b - EVTG_OUT1A input is selected 01_0011b - EVTG_OUT1B input is selected 01_0100b - EVTG_OUT2A input is selected 01_0101b - EVTG_OUT2B input is selected 01_0110b - EVTG_OUT3A input is selected 01_0111b - EVTG_OUT3B input is selected 01_1000b - Reserved 01_1001b - Reserved 01_1010b - LPTMR0 input is selected 01_1011b - LPTMR1 input is selected 01_1100b - SCT Out0 input is selected 01_1101b - SCT Out1 input is selected 01_1110b - SCT Out2 input is selected 01_1111b - SCT Out3 input is selected 10_0000b - SCT Out4 input is selected 10_0001b - SCT Out5 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_0010b - LP_FLEXCOMM0 trigger output 3 input is selected 10_0011b - LP_FLEXCOMM1 trigger output 3 input is selected 10_0100b - LP_FLEXCOMM2 trigger output 3 input is selected 10_0101b - LP_FLEXCOMM3 trigger output 3 input is selected 10_0110b - LP_FLEXCOMM4 trigger output 3 input is selected 10_0111b - LP_FLEXCOMM5 trigger output 3 input is selected 10_1000b - LP_FLEXCOMM6 trigger output 3 input is selected 10_1001b - LP_FLEXCOMM7 trigger output 3 input is selected 10_1010b - LP_FLEXCOMM8 trigger output 3 input is selected 10_1011b - LP_FLEXCOMM9 trigger output 3 input is selected 10_1100b - CMP0_OUT input is selected 10_1101b - CMP1_OUT input is selected 10_1110b - CMP2_OUT input is selected 10_1111b - ENET_PPS_OUT_0 input is selected All other values are reserved.

26.5.1.41 CMP1 Input Connections (CMP1_TRIG)

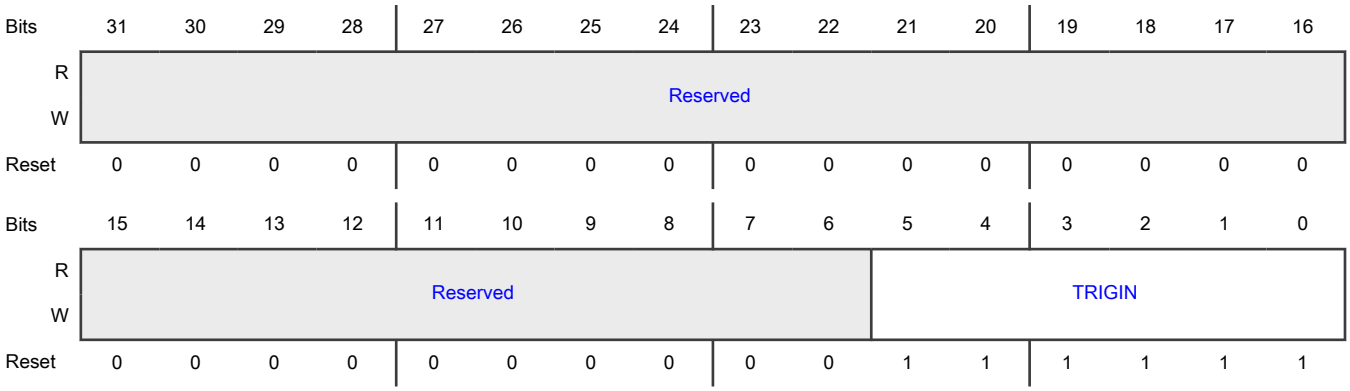
Offset

Register	Offset
CMP1_TRIG	4E0h

Function

This register selects the CMP1 SAMPLE/WINDOW input.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	CMP1 input trigger 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT7 input is selected 00_0010b - SCT0 SCT_OUT4 input is selected 00_0011b - SCT0 SCT_OUT5 input is selected 00_0100b - SCT0 SCT_OUT7 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER3_MAT1 input is selected 00_1001b - CTIMER4_MAT1 input is selected 00_1010b - Reserved 00_1011b - Reserved 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[1] input is selected 00_1110b - ADC1_tcomp[1] input is selected 00_1111b - Reserved 01_0000b - Reserved 01_0001b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 01_0010b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 01_0011b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 01_0100b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 01_0101b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 01_0110b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 01_0111b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 01_1000b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 01_1001b - QDC0_CMP/POS_MATCH input is selected 01_1010b - QDC1_CMP/POS_MATCH input is selected 01_1011b - EVTG_OUT0A input is selected 01_1100b - EVTG_OUT0B input is selected 01_1101b - EVTG_OUT1A input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - EVTG_OUT1B input is selected 01_1111b - EVTG_OUT2A input is selected 10_0000b - EVTG_OUT2B input is selected 10_0001b - EVTG_OUT3A input is selected 10_0010b - EVTG_OUT3B input is selected 10_0011b - LPTMR0 input is selected 10_0100b - LPTMR1 input is selected 10_0101b - GPIO2 Pin Event Trig 0 input is selected 10_0110b - GPIO2 Pin Event Trig 1 input is selected 10_0111b - GPIO3 Pin Event Trig 0 input is selected 10_1000b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.42 CMP2 Input Connections (CMP2_TRIG)

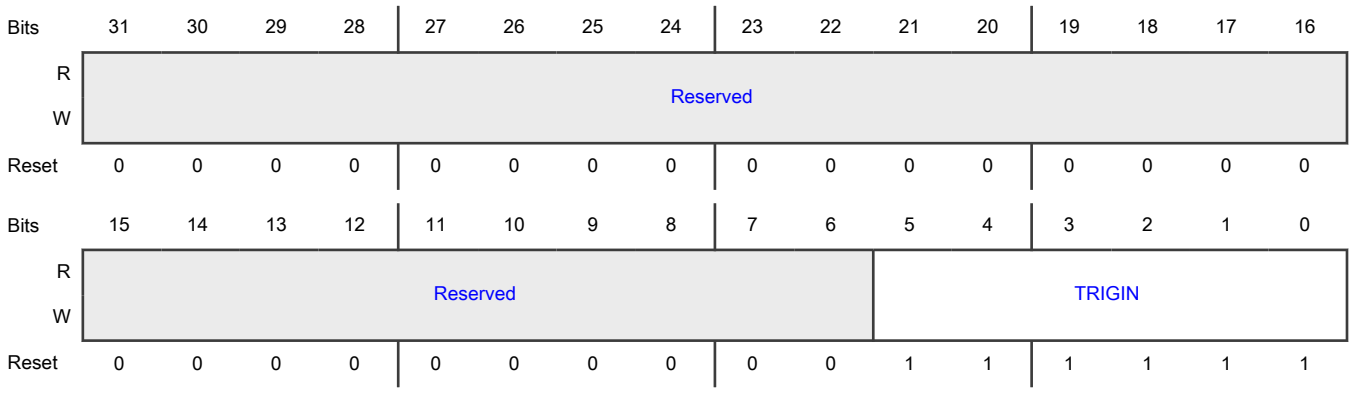
Offset

Register	Offset
CMP2_TRIG	500h

Function

This register selects the CMP2 SAMPLE/WINDOW input.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIGIN	CMP2 input trigger 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT4 input is selected 00_0010b - SCT0 SCT_OUT4 input is selected 00_0011b - SCT0 SCT_OUT5 input is selected 00_0100b - SCT0 SCT_OUT8 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER3_MAT2 input is selected 00_1001b - CTIMER4_MAT2 input is selected 00_1010b - Reserved 00_1011b - Reserved 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[2] input is selected 00_1110b - ADC1_tcomp[2] input is selected 00_1111b - Reserved 01_0000b - Reserved 01_0001b - PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1 input is selected 01_0010b - PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1 input is selected 01_0011b - PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1 input is selected 01_0100b - PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1 input is selected 01_0101b - PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1 input is selected 01_0110b - PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1 input is selected 01_0111b - PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1 input is selected 01_1000b - PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1 input is selected 01_1001b - QDC0_CMP/POS_MATCH input is selected 01_1010b - QDC1_CMP/POS_MATCH input is selected 01_1011b - EVTG_OUT0A input is selected 01_1100b - EVTG_OUT0B input is selected 01_1101b - EVTG_OUT1A input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - EVTG_OUT1B input is selected 01_1111b - EVTG_OUT2A input is selected 10_0000b - EVTG_OUT2B input is selected 10_0001b - EVTG_OUT3A input is selected 10_0010b - EVTG_OUT3B input is selected 10_0011b - LPTMR0 input is selected 10_0100b - LPTMR1 input is selected 10_0101b - GPIO2 Pin Event Trig 0 input is selected 10_0110b - GPIO2 Pin Event Trig 1 input is selected 10_0111b - GPIO3 Pin Event Trig 0 input is selected 10_1000b - GPIO3 Pin Event Trig 1 input is selected All other values are reserved.

26.5.1.43 SINC Filter Channel Trigger Input Connections (SINC_FILTER_CH0 - SINC_FILTER_CH4)

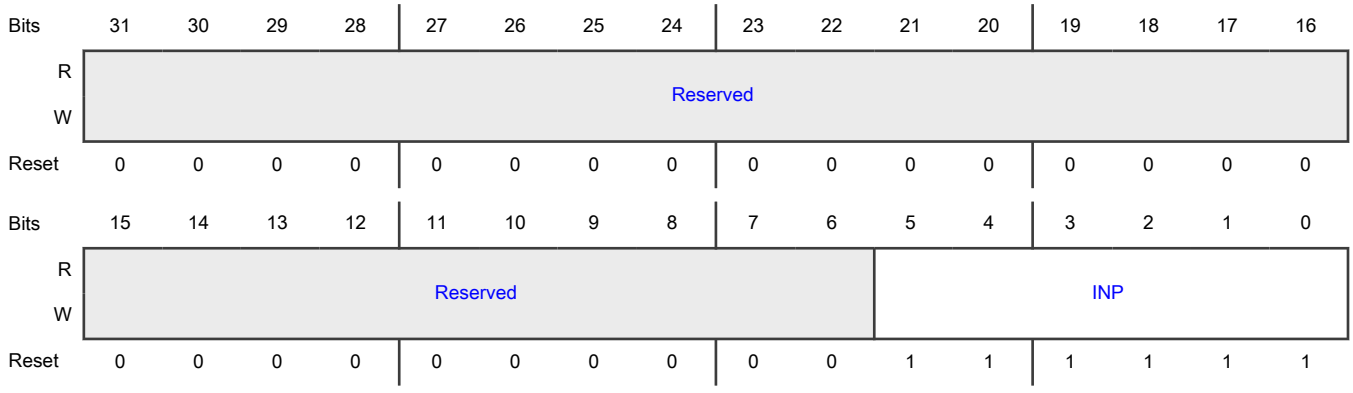
Offset

Register	Offset
SINC_FILTER_CH0	520h
SINC_FILTER_CH1	524h
SINC_FILTER_CH2	528h
SINC_FILTER_CH3	52Ch
SINC_FILTER_CH4	530h

Function

This register is used to select the SINC Filter channel trigger inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	SINC FILTER trigger input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT1 input is selected 00_0010b - SCT_OUT4 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT9 input is selected 00_0101b - CTIMER0_MAT3 input is selected 00_0110b - CTIMER1_MAT3 input is selected 00_0111b - CTIMER2_MAT3 input is selected 00_1000b - CTIMER3_MAT3 input is selected 00_1001b - CTIMER4_MAT3 input is selected 00_1010b - Reserved 00_1011b - Reserved 00_1100b - PINT GPIO_INT_BMAT input is selected 00_1101b - ADC0_tcomp[0] input is selected 00_1110b - ADC0_tcomp[1] input is selected 00_1111b - ADC0_tcomp[2] input is selected 01_0000b - ADC0_tcomp[3] input is selected 01_0001b - ADC1_tcomp[0] input is selected 01_0010b - ADC1_tcomp[1] input is selected 01_0011b - ADC1_tcomp[2] input is selected 01_0100b - ADC1_tcomp[3] input is selected 01_0101b - CMP0_OUT input is selected 01_0110b - CMP1_OUT input is selected 01_0111b - CMP2_OUT input is selected 01_1000b - PWM0_SM0_MUX_TRIG0 input is selected 01_1001b - PWM0_SM0_MUX_TRIG1 input is selected 01_1010b - PWM0_SM1_MUX_TRIG0 input is selected 01_1011b - PWM0_SM1_MUX_TRIG1 input is selected 01_1100b - PWM0_SM2_MUX_TRIG0 input is selected 01_1101b - PWM0_SM2_MUX_TRIG1 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - PWM0_SM3_MUX_TRIG0 input is selected 01_1111b - PWM0_SM3_MUX_TRIG1 input is selected 10_0000b - PWM1_SM0_MUX_TRIG0 input is selected 10_0001b - PWM1_SM0_MUX_TRIG1 input is selected 10_0010b - PWM1_SM1_MUX_TRIG0 input is selected 10_0011b - PWM1_SM1_MUX_TRIG1 input is selected 10_0100b - PWM1_SM2_MUX_TRIG0 input is selected 10_0101b - PWM1_SM2_MUX_TRIG1 input is selected 10_0110b - PWM1_SM3_MUX_TRIG0 input is selected 10_0111b - PWM1_SM3_MUX_TRIG1 input is selected 10_1000b - QDC0_CMP/POS_MATCH input is selected 10_1001b - QDC1_CMP/POS_MATCH input is selected 10_1010b - EVTG_OUT0A input is selected 10_1011b - EVTG_OUT0B input is selected 10_1100b - EVTG_OUT1A input is selected 10_1101b - EVTG_OUT1B input is selected 10_1110b - EVTG_OUT2A input is selected 10_1111b - EVTG_OUT2B input is selected 11_0000b - EVTG_OUT3A input is selected 11_0001b - EVTG_OUT3B input is selected 11_0010b - LPTMR0 input is selected 11_0011b - LPTMR1 input is selected 11_0100b - FlexIO CH0 input is selected 11_0101b - FlexIO CH1 input is selected 11_0110b - FlexIO CH2 input is selected 11_0111b - FlexIO CH3 input is selected 11_1000b - WUU input is selected All other values are reserved.

26.5.1.44 OPAMP Trigger Input Connections (OPAMP0_TRIG - OPAMP2_TRIG)

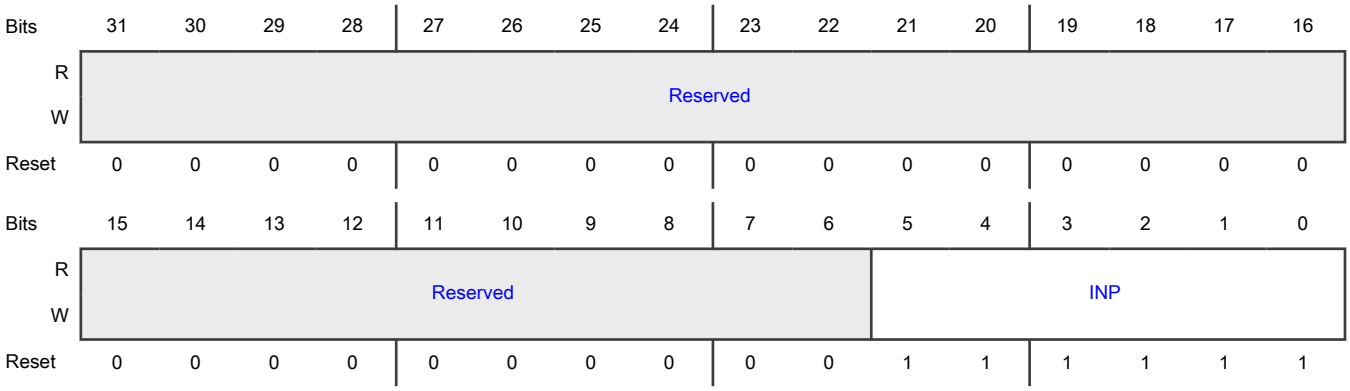
Offset

Register	Offset
OPAMP0_TRIG	580h
OPAMP1_TRIG	584h
OPAMP2_TRIG	588h

Function

This register is used to select the OPAMP trigger inputs.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 INP	OPAMP trigger input connections 00_0000b - PINT PIN_INT0 input is selected 00_0001b - PINT PIN_INT1 input is selected 00_0010b - PINT PIN_INT2 input is selected 00_0011b - PINT PIN_INT3 input is selected 00_0100b - SCT_OUT4 input is selected 00_0101b - SCT_OUT5 input is selected 00_0110b - SCT_OUT6 input is selected 00_0111b - SCT_OUT7 input is selected 00_1000b - SCT_OUT8 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_1001b - CTIMER0_MAT3 input is selected 00_1010b - CTIMER1_MAT3 input is selected 00_1011b - CTIMER2_MAT3 input is selected 00_1100b - CTIMER3_MAT3 input is selected 00_1101b - CTIMER4_MAT3 input is selected 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - ADC0_tcomp[0] input is selected 01_0000b - ADC0_tcomp[1] input is selected 01_0001b - ADC0_tcomp[2] input is selected 01_0010b - ADC0_tcomp[3] input is selected 01_0011b - ADC1_tcomp[0] input is selected 01_0100b - ADC1_tcomp[1] input is selected 01_0101b - ADC1_tcomp[2] input is selected 01_0110b - ADC1_tcomp[3] input is selected 01_0111b - PWM0_SM0_MUX_TRIG0 input is selected 01_1000b - PWM0_SM0_MUX_TRIG1 input is selected 01_1001b - PWM0_SM1_MUX_TRIG0 input is selected 01_1010b - PWM0_SM1_MUX_TRIG1 input is selected 01_1011b - PWM0_SM2_MUX_TRIG0 input is selected 01_1100b - PWM0_SM2_MUX_TRIG1 input is selected 01_1101b - PWM0_SM3_MUX_TRIG0 input is selected 01_1110b - PWM0_SM3_MUX_TRIG1 input is selected 01_1111b - PWM1_SM0_MUX_TRIG0 input is selected 10_0000b - PWM1_SM0_MUX_TRIG1 input is selected 10_0001b - PWM1_SM1_MUX_TRIG0 input is selected 10_0010b - PWM1_SM1_MUX_TRIG1 input is selected 10_0011b - PWM1_SM2_MUX_TRIG0 input is selected 10_0100b - PWM1_SM2_MUX_TRIG1 input is selected 10_0101b - PWM1_SM3_MUX_TRIG0 input is selected 10_0110b - PWM1_SM3_MUX_TRIG1 input is selected 10_0111b - EVTG_OUT0A input is selected 10_1000b - EVTG_OUT0B input is selected 10_1001b - EVTG_OUT1A input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10_1010b - EVTG_OUT1B input is selected 10_1011b - EVTG_OUT2A input is selected 10_1100b - EVTG_OUT2B input is selected 10_1101b - EVTG_OUT3A input is selected 10_1110b - EVTG_OUT3B input is selected 10_1111b - TRIG_IN0 input is selected 11_0000b - TRIG_IN1 input is selected 11_0001b - TRIG_IN2 input is selected 11_0010b - TRIG_IN3 input is selected 11_0011b - FlexIO CH4 input is selected 11_0100b - FlexIO CH5 input is selected 11_0101b - FlexIO CH6 input is selected 11_0110b - FlexIO CH7 input is selected All other values are reserved.

26.5.1.45 LP_FLEXCOMM0 Trigger Input Connections (FLEXCOMM0_TRIG)

Offset

Register	Offset
FLEXCOMM0_TRIG	5A0h

Function

This register is used to select the LP_FLEXCOMM0 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM0 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT6 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT6 input is selected 00_0101b - SCT_OUT7 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - CTIMER4_MAT0 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.46 LP_FLEXCOMM1 Trigger Input Connections (FLEXCOMM1_TRIG)

Offset

Register	Offset
FLEXCOMM1_TRIG	5C0h

Function

This register is used to select the LP_FLEXCOMM1 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM1 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT6 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT6 input is selected 00_0101b - SCT_OUT7 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - CTIMER4_MAT0 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.47 LP_FLEXCOMM2 Trigger Input Connections (FLEXCOMM2_TRIG)

Offset

Register	Offset
FLEXCOMM2_TRIG	5E0h

Function

This register is used to select the LP_FLEXCOMM2 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM2 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT6 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT8 input is selected 00_0101b - SCT_OUT9 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT1 input is selected 00_1001b - CTIMER3_MAT1 input is selected 00_1010b - CTIMER4_MAT1 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.48 LP_FLEXCOMM3 Trigger Input Connections (FLEXCOMM3_TRIG)

Offset

Register	Offset
FLEXCOMM3_TRIG	600h

Function

This register is used to select the LP_FLEXCOMM3 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM3 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT5 input is selected 00_0100b - SCT_OUT8 input is selected 00_0101b - SCT_OUT9 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT1 input is selected 00_1001b - CTIMER3_MAT1 input is selected 00_1010b - CTIMER4_MAT1 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.49 LP_FLEXCOMM4 Trigger Input Connections (FLEXCOMM4_TRIG)

Offset

Register	Offset
FLEXCOMM4_TRIG	620h

Function

This register is used to select the LP_FLEXCOMM4 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM4 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT0 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - SCT_OUT2 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT2 input is selected 00_1001b - CTIMER3_MAT2 input is selected 00_1010b - CTIMER4_MAT2 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.50 LP_FLEXCOMM5 Trigger Input Connections (FLEXCOMM5_TRIG)

Offset

Register	Offset
FLEXCOMM5_TRIG	640h

Function

This register is used to select the LP_FLEXCOMM5 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM5 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT0 input is selected 00_0100b - SCT_OUT1 input is selected 00_0101b - SCT_OUT2 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT2 input is selected 00_1001b - CTIMER3_MAT2 input is selected 00_1010b - CTIMER4_MAT2 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.51 LP_FLEXCOMM6 Trigger Input Connections (FLEXCOMM6_TRIG)

Offset

Register	Offset
FLEXCOMM6_TRIG	660h

Function

This register is used to select the LP_FLEXCOMM6 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM6 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT0 input is selected 00_0100b - SCT_OUT3 input is selected 00_0101b - SCT_OUT4 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT3 input is selected 00_1001b - CTIMER3_MAT3 input is selected 00_1010b - CTIMER4_MAT3 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.52 LP_FLEXCOMM7 Trigger Input Connections (FLEXCOMM7_TRIG)

Offset

Register	Offset
FLEXCOMM7_TRIG	680h

Function

This register is used to select the LP_FLEXCOMM7 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM7 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT0 input is selected 00_0100b - SCT_OUT3 input is selected 00_0101b - SCT_OUT4 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT3 input is selected 00_1001b - CTIMER3_MAT3 input is selected 00_1010b - CTIMER4_MAT3 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.53 LP_FLEXCOMM8 Trigger Input Connections (FLEXCOMM8_TRIG)

Offset

Register	Offset
FLEXCOMM8_TRIG	6A0h

Function

This register is used to select the LP_FLEXCOMM8 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM8 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT0 input is selected 00_0100b - SCT_OUT3 input is selected 00_0101b - SCT_OUT4 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT3 input is selected 00_1001b - CTIMER3_MAT3 input is selected 00_1010b - CTIMER4_MAT3 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.54 LP_FLEXCOMM9 Trigger Input Connections (FLEXCOMM9_TRIG)

Offset

Register	Offset
FLEXCOMM9_TRIG	6C0h

Function

This register is used to select the LP_FLEXCOMM9 trigger inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Function
31-6 —	Reserved
5-0 INP	LP_FLEXCOMM9 trigger input connections 00_0000b - PINT PIN_INT4 input is selected 00_0001b - PINT PIN_INT5 input is selected 00_0010b - PINT PIN_INT7 input is selected 00_0011b - SCT_OUT0 input is selected 00_0100b - SCT_OUT3 input is selected 00_0101b - SCT_OUT4 input is selected 00_0110b - CTIMER0_MAT1 input is selected 00_0111b - CTIMER1_MAT1 input is selected 00_1000b - CTIMER2_MAT0 input is selected 00_1001b - CTIMER3_MAT0 input is selected 00_1010b - CTIMER4_MAT0 input is selected 00_1011b - LPTMR0 input is selected 00_1100b - LPTMR1 input is selected 00_1101b - Reserved 00_1110b - PINT GPIO_INT_BMAT input is selected 00_1111b - CMP0_OUT input is selected 01_0000b - CMP1_OUT input is selected 01_0001b - CMP2_OUT input is selected 01_0010b - EVTG_OUT0A input is selected 01_0011b - EVTG_OUT0B input is selected 01_0100b - EVTG_OUT1A input is selected 01_0101b - EVTG_OUT1B input is selected 01_0110b - EVTG_OUT2A input is selected 01_0111b - EVTG_OUT2B input is selected 01_1000b - EVTG_OUT3A input is selected 01_1001b - EVTG_OUT3B input is selected 01_1010b - TRIG_IN0 input is selected 01_1011b - TRIG_IN1 input is selected 01_1100b - TRIG_IN2 input is selected 01_1101b - TRIG_IN3 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1110b - TRIG_IN4 input is selected 01_1111b - TRIG_IN10 input is selected 10_0000b - TRIG_IN11 input is selected 10_0001b - FlexIO CH4 input is selected 10_0010b - FlexIO CH5 input is selected 10_0011b - FlexIO CH6 input is selected 10_0100b - FlexIO CH7 input is selected 10_0101b - USB0 ipp_ind_uart_rxd_usbmux input is selected 10_0110b - GPIO2 Pin Event Trig 0 input is selected 10_0111b - GPIO2 Pin Event Trig 1 input is selected 10_1000b - GPIO3 Pin Event Trig 0 input is selected 10_1001b - GPIO3 Pin Event Trig 1 input is selected 10_1010b - WUU input is selected All other values are reserved.

26.5.1.55 FlexIO Trigger Input Connections (FLEXIO_TRIG0 - FLEXIO_TRIG7)

Offset

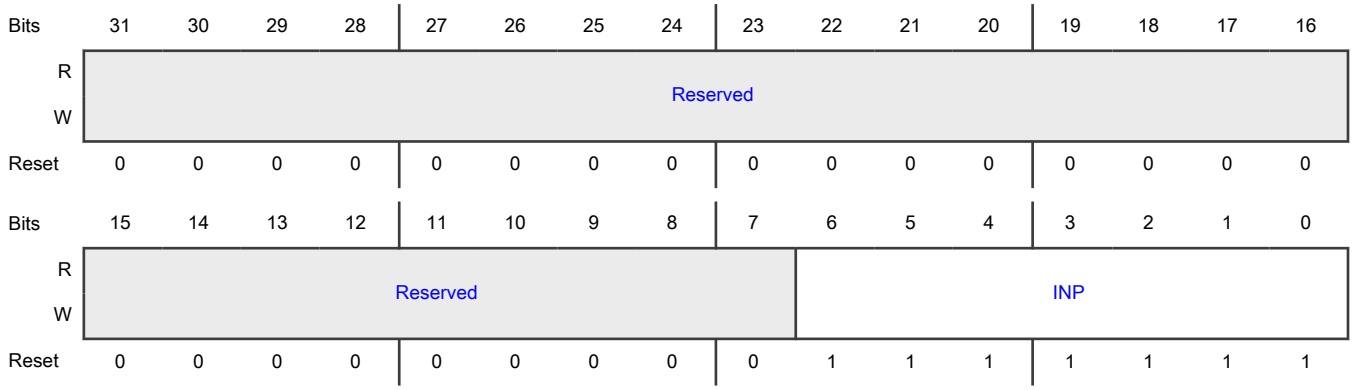
For a = 0 to 7:

Register	Offset
FLEXIO_TRIGa	6E0h + (a × 4h)

Function

This register is used to select the FlexIO trigger inputs.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 INP	Input number for FlexIO0. 000_0000b - PINT PIN_INT4 input is selected 000_0001b - PINT PIN_INT5 input is selected 000_0010b - PINT PIN_INT6 input is selected 000_0011b - PINT PIN_INT7 input is selected 000_0100b - SCT_OUT5 input is selected 000_0101b - SCT_OUT6 input is selected 000_0110b - SCT_OUT7 input is selected 000_0111b - SCT_OUT8 input is selected 000_1000b - SCT_OUT9 input is selected 000_1001b - T0_MAT1 input is selected 000_1010b - T1_MAT1 input is selected 000_1011b - T2_MAT1 input is selected 000_1100b - T3_MAT1 input is selected 000_1101b - T4_MAT1 input is selected 000_1110b - LPTMR0 input is selected 000_1111b - LPTMR1 input is selected 001_0000b - Reserved 001_0001b - PINT GPIO_INT_BMAT input is selected 001_0010b - ADC0_tcomp[0] input is selected 001_0011b - ADC0_tcomp[1] input is selected 001_0100b - ADC0_tcomp[2] input is selected 001_0101b - ADC0_tcomp[3] input is selected 001_0110b - ADC1_tcomp[0] input is selected 001_0111b - ADC1_tcomp[1] input is selected 001_1000b - ADC1_tcomp[2] input is selected 001_1001b - ADC1_tcomp[3] input is selected 001_1010b - CMP0_OUT input is selected 001_1011b - CMP1_OUT input is selected 001_1100b - CMP2_OUT input is selected 001_1101b - PWM0_SM0_MUX_TRIG0 input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001_1110b - PWM0_SM0_MUX_TRIG1 input is selected 001_1111b - PWM0_SM1_MUX_TRIG0 input is selected 010_0000b - PWM0_SM1_MUX_TRIG1 input is selected 010_0001b - PWM0_SM2_MUX_TRIG0 input is selected 010_0010b - PWM0_SM2_MUX_TRIG1 input is selected 010_0011b - PWM0_SM3_MUX_TRIG0 input is selected 010_0100b - PWM0_SM3_MUX_TRIG1 input is selected 010_0101b - PWM1_SM0_MUX_TRIG0 input is selected 010_0110b - PWM1_SM0_MUX_TRIG1 input is selected 010_0111b - PWM1_SM1_MUX_TRIG0 input is selected 010_1000b - PWM1_SM1_MUX_TRIG1 input is selected 010_1001b - PWM1_SM2_MUX_TRIG0 input is selected 010_1010b - PWM1_SM2_MUX_TRIG1 input is selected 010_1011b - PWM1_SM3_MUX_TRIG0 input is selected 010_1100b - PWM1_SM3_MUX_TRIG1 input is selected 010_1101b - EVTG_OUT0A input is selected 010_1110b - EVTG_OUT0B input is selected 010_1111b - EVTG_OUT1A input is selected 011_0000b - EVTG_OUT1B input is selected 011_0001b - EVTG_OUT2A input is selected 011_0010b - EVTG_OUT2B input is selected 011_0011b - EVTG_OUT3A input is selected 011_0100b - EVTG_OUT3B input is selected 011_0101b - TRIG_IN0 input is selected 011_0110b - TRIG_IN1 input is selected 011_0111b - TRIG_IN2 input is selected 011_1000b - TRIG_IN3 input is selected 011_1001b - TRIG_IN4 input is selected 011_1010b - SINC Filter CH0 Conversion Complete input is selected 011_1011b - SINC Filter CH1 Conversion Complete input is selected 011_1100b - SINC Filter CH2 Conversion Complete input is selected 011_1101b - SINC Filter CH3 Conversion Complete input is selected 011_1110b - SINC Filter CH4 Conversion Complete input is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011_1111b - LP_FLEXCOMM0 trig 0 (lpuart_trg_txword) input is selected 100_0000b - LP_FLEXCOMM0 trig 1 (lpuart_trg_rxword) input is selected 100_0001b - LP_FLEXCOMM0 trig 2 (lpuart_trg_rxidle) input is selected 100_0010b - LP_FLEXCOMM1 trig 0 input is selected 100_0011b - LP_FLEXCOMM1 trig 1 input is selected 100_0100b - LP_FLEXCOMM1 trig 2 input is selected 100_0101b - LP_FLEXCOMM2 trig 0 input is selected 100_0110b - LP_FLEXCOMM2 trig 1 input is selected 100_0111b - LP_FLEXCOMM2 trig 2 input is selected 100_1000b - LP_FLEXCOMM3 trig 0 input is selected 100_1001b - LP_FLEXCOMM3 trig 1 input is selected 100_1010b - LP_FLEXCOMM3 trig 2 input is selected 100_1011b - LP_FLEXCOMM3 trig 3 input is selected 100_1100b - WUU input is selected All other values are reserved.

26.5.1.56 DMA0 Request Enable0 (DMA0_REQ_ENABLE0)

Offset

Register	Offset
DMA0_REQ_ENABLE0	700h

Function

DMA request 0-31 enable for DMA0. One bit per request. 0: DMA request to DMA0 and response from DMA0 are blocked. 1:DMA request and response are enabled for DMA0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN0	REQ8_EN0	REQ7_EN0	REQ6_EN0	REQ5_EN0	REQ4_EN0	REQ3_EN0	REQ2_EN0	REQ1_EN0	0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 REQ31_EN0	This register is used to enable and disable EVTG0 OUT0A request. 0b - Disable 1b - Enable
30 REQ30_EN0	This register is used to enable and disable CMP2 DMA_request. 0b - Disable 1b - Enable
29 REQ29_EN0	This register is used to enable and disable CMP1 DMA_request. 0b - Disable 1b - Enable
28 REQ28_EN0	This register is used to enable and disable CMP0 DMA_request. 0b - Disable 1b - Enable
27 REQ27_EN0	This register is used to enable and disable DAC2 FIFO_request. 0b - Disable 1b - Enable
26 REQ26_EN0	This register is used to enable and disable DAC1 FIFO_request. 0b - Disable 1b - Enable
25 REQ25_EN0	This register is used to enable and disable DAC0 FIFO_request. 0b - Disable 1b - Enable
24 REQ24_EN0	This register is used to enable and disable ADC1 FIFO B request. 0b - Disable 1b - Enable
23 REQ23_EN0	This register is used to enable and disable ADC1 FIFO A request. 0b - Disable 1b - Enable
22 REQ22_EN0	This register is used to enable and disable ADC0 FIFO B request. 0b - Disable 1b - Enable
21 REQ21_EN0	This register is used to enable and disable ADC0 FIFO A request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
20 REQ20_EN0	This register is used to enable and disable SCT0 DMA1 request. 0b - Disable 1b - Enable
19 REQ19_EN0	This register is used to enable and disable SCT0 DMA0 request. 0b - Disable 1b - Enable
18 REQ18_EN0	This register is used to enable and disable MICFIL0 FIFO_request. 0b - Disable 1b - Enable
17 REQ17_EN0	This register is used to enable and disable WUU0 wake up event request. 0b - Disable 1b - Enable
16 REQ16_EN0	This register is used to enable and disable CTIMER4 DMAREQ_M1 request. 0b - Disable 1b - Enable
15 REQ15_EN0	This register is used to enable and disable CTIMER4 DMAREQ_M0 request. 0b - Disable 1b - Enable
14 REQ14_EN0	This register is used to enable and disable CTIMER3 DMAREQ_M1 request. 0b - Disable 1b - Enable
13 REQ13_EN0	This register is used to enable and disable CTIMER3 DMAREQ_M0 request. 0b - Disable 1b - Enable
12 REQ12_EN0	This register is used to enable and disable CTIMER2 DMAREQ_M1 request. 0b - Disable 1b - Enable
11 REQ11_EN0	This register is used to enable and disable CTIMER2 DMAREQ_M0 request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 REQ10_EN0	This register is used to enable and disable CTIMER1 DMAREQ_M1 request. 0b - Disable 1b - Enable
9 REQ9_EN0	This register is used to enable and disable CTIMER1 DMAREQ_M0 request. 0b - Disable 1b - Enable
8 REQ8_EN0	This register is used to enable and disable CTIMER0 DMAREQ_M1 request. 0b - Disable 1b - Enable
7 REQ7_EN0	This register is used to enable and disable CTIMER0 DMAREQ_M0 request. 0b - Disable 1b - Enable
6 REQ6_EN0	This register is used to enable and disable PINT0 INT3 request. 0b - Disable 1b - Enable
5 REQ5_EN0	This register is used to enable and disable PINT0 INT2 request. 0b - Disable 1b - Enable
4 REQ4_EN0	This register is used to enable and disable PINT0 INT1 request. 0b - Disable 1b - Enable
3 REQ3_EN0	This register is used to enable and disable PINT0 INT0 request. 0b - Disable 1b - Enable
2 REQ2_EN0	This register is used to enable and disable FLEXSPI0 transmit event request. 0b - Disable 1b - Enable
1 REQ1_EN0	This register is used to enable and disable FLEXSPI0 receive event request. 0b - Disable 1b - Enable
0	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	

26.5.1.57 DMA0 Request Enable0 (DMA0_REQ_ENABLE0_SET)

Offset

Register	Offset
DMA0_REQ_ENABLE0_SET	704h

Function

Writing a 1 to a bit in this register sets the corresponding bit in DMA0_REQ_ENABLE0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																0
W	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN0	REQ8_EN0	REQ7_EN0	REQ6_EN0	REQ5_EN0	REQ4_EN0	REQ3_EN0	REQ2_EN0	REQ1_EN0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ31_EN0	Writing a 1 to REQ31_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
30 REQ30_EN0	Writing a 1 to REQ30_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
29 REQ29_EN0	Writing a 1 to REQ29_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
28	Writing a 1 to REQ28_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ28_EN0	
27 REQ27_EN0	Writing a 1 to REQ27_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
26 REQ26_EN0	Writing a 1 to REQ26_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
25 REQ25_EN0	Writing a 1 to REQ25_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
24 REQ24_EN0	Writing a 1 to REQ24_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
23 REQ23_EN0	Writing a 1 to REQ23_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
22 REQ22_EN0	Writing a 1 to REQ22_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
21 REQ21_EN0	Writing a 1 to REQ21_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
20 REQ20_EN0	Writing a 1 to REQ20_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
19 REQ19_EN0	Writing a 1 to REQ19_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
18 REQ18_EN0	Writing a 1 to REQ18_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
17 REQ17_EN0	Writing a 1 to REQ17_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
16 REQ16_EN0	Writing a 1 to REQ16_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
15 REQ15_EN0	Writing a 1 to REQ15_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
14 REQ14_EN0	Writing a 1 to REQ14_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 REQ13_EN0	Writing a 1 to REQ13_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
12 REQ12_EN0	Writing a 1 to REQ12_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
11 REQ11_EN0	Writing a 1 to REQ11_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
10 REQ10_EN0	Writing a 1 to REQ10_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
9 REQ9_EN0	Writing a 1 to REQ9_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
8 REQ8_EN0	Writing a 1 to REQ8_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
7 REQ7_EN0	Writing a 1 to REQ7_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
6 REQ6_EN0	Writing a 1 to REQ6_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
5 REQ5_EN0	Writing a 1 to REQ5_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
4 REQ4_EN0	Writing a 1 to REQ4_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
3 REQ3_EN0	Writing a 1 to REQ3_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
2 REQ2_EN0	Writing a 1 to REQ2_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
1 REQ1_EN0	Writing a 1 to REQ1_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE0.
0 —	Reserved

26.5.1.58 DMA0 Request Enable0 (DMA0_REQ_ENABLE0_CLR)

Offset

Register	Offset
DMA0_REQ_ENABLE0_CLR	708h

Function
Writing a 1 to a bit in this register clears the corresponding bit in DMA0_REQ_ENABLE0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																0
W	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN0	REQ8_EN0	REQ7_EN0	REQ6_EN0	REQ5_EN0	REQ4_EN0	REQ3_EN0	REQ2_EN0	REQ1_EN0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ31_EN0	Writing a 1 to REQ31_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
30 REQ30_EN0	Writing a 1 to REQ30_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
29 REQ29_EN0	Writing a 1 to REQ29_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
28 REQ28_EN0	Writing a 1 to REQ28_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
27 REQ27_EN0	Writing a 1 to REQ27_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
26	Writing a 1 to REQ26_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ26_EN0	
25 REQ25_EN0	Writing a 1 to REQ25_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
24 REQ24_EN0	Writing a 1 to REQ24_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
23 REQ23_EN0	Writing a 1 to REQ23_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
22 REQ22_EN0	Writing a 1 to REQ22_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
21 REQ21_EN0	Writing a 1 to REQ21_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
20 REQ20_EN0	Writing a 1 to REQ20_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
19 REQ19_EN0	Writing a 1 to REQ19_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
18 REQ18_EN0	Writing a 1 to REQ18_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
17 REQ17_EN0	Writing a 1 to REQ17_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
16 REQ16_EN0	Writing a 1 to REQ16_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
15 REQ15_EN0	Writing a 1 to REQ15_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
14 REQ14_EN0	Writing a 1 to REQ14_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
13 REQ13_EN0	Writing a 1 to REQ13_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
12 REQ12_EN0	Writing a 1 to REQ12_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 REQ11_EN0	Writing a 1 to REQ11_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
10 REQ10_EN0	Writing a 1 to REQ10_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
9 REQ9_EN0	Writing a 1 to REQ9_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
8 REQ8_EN0	Writing a 1 to REQ8_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
7 REQ7_EN0	Writing a 1 to REQ7_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
6 REQ6_EN0	Writing a 1 to REQ6_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
5 REQ5_EN0	Writing a 1 to REQ5_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
4 REQ4_EN0	Writing a 1 to REQ4_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
3 REQ3_EN0	Writing a 1 to REQ3_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
2 REQ2_EN0	Writing a 1 to REQ2_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
1 REQ1_EN0	Writing a 1 to REQ1_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE0.
0 —	Reserved

26.5.1.59 DMA0 Request Enable0 (DMA0_REQ_ENABLE0_TOG)

Offset

Register	Offset
DMA0_REQ_ENABLE0_TOG	70Ch

Function

Writing a 1 to a bit in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																0
W	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN0	REQ8_EN0	REQ7_EN0	REQ6_EN0	REQ5_EN0	REQ4_EN0	REQ3_EN0	REQ2_EN0	REQ1_EN0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ31_EN0	Writing a 1 to REQ31_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
30 REQ30_EN0	Writing a 1 to REQ30_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
29 REQ29_EN0	Writing a 1 to REQ29_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
28 REQ28_EN0	Writing a 1 to REQ28_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
27 REQ27_EN0	Writing a 1 to REQ27_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
26 REQ26_EN0	Writing a 1 to REQ26_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
25 REQ25_EN0	Writing a 1 to REQ25_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
24 REQ24_EN0	Writing a 1 to REQ24_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
23	Writing a 1 to REQ23_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ23_EN0	
22 REQ22_EN0	Writing a 1 to REQ22_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
21 REQ21_EN0	Writing a 1 to REQ21_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
20 REQ20_EN0	Writing a 1 to REQ20_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
19 REQ19_EN0	Writing a 1 to REQ19_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
18 REQ18_EN0	Writing a 1 to REQ18_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
17 REQ17_EN0	Writing a 1 to REQ17_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
16 REQ16_EN0	Writing a 1 to REQ16_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
15 REQ15_EN0	Writing a 1 to REQ15_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
14 REQ14_EN0	Writing a 1 to REQ14_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
13 REQ13_EN0	Writing a 1 to REQ13_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
12 REQ12_EN0	Writing a 1 to REQ12_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
11 REQ11_EN0	Writing a 1 to REQ11_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
10 REQ10_EN0	Writing a 1 to REQ10_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
9 REQ9_EN0	Writing a 1 to RE9_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 REQ8_EN0	Writing a 1 to REQ8_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
7 REQ7_EN0	Writing a 1 to REQ7_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
6 REQ6_EN0	Writing a 1 to REQ6_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
5 REQ5_EN0	Writing a 1 to REQ5_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
4 REQ4_EN0	Writing a 1 to REQ4_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
3 REQ3_EN0	Writing a 1 to REQ3_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
2 REQ2_EN0	Writing a 1 to REQ2_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
1 REQ1_EN0	Writing a 1 to REQ1_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE0.
0 —	Reserved

26.5.1.60 DMA0 Request Enable1 (DMA0_REQ_ENABLE1)

Offset

Register	Offset
DMA0_REQ_ENABLE1	710h

Function

DMA request 32-63 enable for DMA0. One bit per request. 0: DMA request to DMA0 and response from DMA0 are blocked. 1:DMA request and response are enabled for DMA0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...	0	0	REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 REQ63_EN0	This register is used to enable and disable FlexIO0 Shifter2 Status DMA request OR Timer2 Status DMA request. 0b - Disable 1b - Enable
30 REQ62_EN0	This register is used to enable and disable FlexIO0 Shifter1 Status DMA request OR Timer1 Status DMA request. 0b - Disable 1b - Enable
29 REQ61_EN0	This register is used to enable and disable FlexIO0 Shifter0 Status DMA request OR Timer0 Status DMA request. 0b - Disable 1b - Enable
28 REQ60_EN0	This register is used to enable and disable CAN1 DMA request. 0b - Disable 1b - Enable
27 REQ59_EN0	This register is used to enable and disable CAN0 DMA request. 0b - Disable 1b - Enable
26 REQ58_EN0	This register is used to enable and disable LPTMR1 counter match event request. 0b - Disable 1b - Enable
25 REQ57_EN0	This register is used to enable and disable LPTMR0 counter match event request. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
24 —	Reserved
23 —	Reserved
22 REQ54_EN0	This register is used to enable and disable PWM1 Req_val3 request. 0b - Disable 1b - Enable
21 REQ53_EN0	This register is used to enable and disable PWM1 Req_val2 request. 0b - Disable 1b - Enable
20 REQ52_EN0	This register is used to enable and disable PWM1 Req_val1 request. 0b - Disable 1b - Enable
19 REQ51_EN0	This register is used to enable and disable PWM1 Req_val0 request. 0b - Disable 1b - Enable
18 REQ50_EN0	This register is used to enable and disable PWM1 Req_capt3 request. 0b - Disable 1b - Enable
17 REQ49_EN0	This register is used to enable and disable PWM1 Req_capt2 request. 0b - Disable 1b - Enable
16 REQ48_EN0	This register is used to enable and disable PWM1 Req_capt1 request. 0b - Disable 1b - Enable
15 REQ47_EN0	This register is used to enable and disable PWM1 Req_capt0 request. 0b - Disable 1b - Enable
14 REQ46_EN0	This register is used to enable and disable PWM0 Req_val3 request. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
13 REQ45_EN0	This register is used to enable and disable PWM0 Req_val2 request. 0b - Disable 1b - Enable
12 REQ44_EN0	This register is used to enable and disable PWM0 Req_val1 request. 0b - Disable 1b - Enable
11 REQ43_EN0	This register is used to enable and disable PWM0 Req_val0 request. 0b - Disable 1b - Enable
10 REQ42_EN0	This register is used to enable and disable PWM0 Req_capt3 request. 0b - Disable 1b - Enable
9 REQ41_EN0	This register is used to enable and disable PWM0 Req_capt2 request. 0b - Disable 1b - Enable
8 REQ40_EN0	This register is used to enable and disable PWM0 Req_capt1 request. 0b - Disable 1b - Enable
7 REQ39_EN0	This register is used to enable and disable PWM0 Req_capt0 request. 0b - Disable 1b - Enable
6 REQ38_EN0	This register is used to enable and disable EVTG0 OUT3B request. 0b - Disable 1b - Enable
5 REQ37_EN0	This register is used to enable and disable EVTG0 OUT3A request. 0b - Disable 1b - Enable
4 REQ36_EN0	This register is used to enable and disable EVTG0 OUT2B request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 REQ35_EN0	This register is used to enable and disable EVTG0 OUT2A request. 0b - Disable 1b - Enable
2 REQ34_EN0	This register is used to enable and disable EVTG0 OUT1B request. 0b - Disable 1b - Enable
1 REQ33_EN0	This register is used to enable and disable EVTG0 OUT1A request. 0b - Disable 1b - Enable
0 REQ32_EN0	This register is used to enable and disable EVTG0 OUT0B request. 0b - Disable 1b - Enable

26.5.1.61 DMA0 Request Enable1 (DMA0_REQ_ENABLE1_SET)

Offset

Register	Offset
DMA0_REQ_ENABLE1_SET	714h

Function

Writing a 1 to REQ_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								0	0							
W	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...			REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ63_EN0	Writing a 1 to REQ63_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
30 REQ62_EN0	Writing a 1 to REQ62_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
29 REQ61_EN0	Writing a 1 to REQ61_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
28 REQ60_EN0	Writing a 1 to REQ60_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
27 REQ59_EN0	Writing a 1 to REQ59_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
26 REQ58_EN0	Writing a 1 to REQ58_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
25 REQ57_EN0	Writing a 1 to REQ57_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
24 —	Reserved
23 —	Reserved
22 REQ54_EN0	Writing a 1 to REQ54_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
21 REQ53_EN0	Writing a 1 to REQ53_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
20 REQ52_EN0	Writing a 1 to REQ52_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
19 REQ51_EN0	Writing a 1 to REQ51_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
18 REQ50_EN0	Writing a 1 to REQ50_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
17	Writing a 1 to REQ49_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ49_EN0	
16 REQ48_EN0	Writing a 1 to REQ48_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
15 REQ47_EN0	Writing a 1 to REQ47_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
14 REQ46_EN0	Writing a 1 to REQ46_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
13 REQ45_EN0	Writing a 1 to REQ45_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
12 REQ44_EN0	Writing a 1 to REQ44_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
11 REQ43_EN0	Writing a 1 to REQ43_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
10 REQ42_EN0	Writing a 1 to REQ42_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
9 REQ41_EN0	Writing a 1 to REQ41_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
8 REQ40_EN0	Writing a 1 to REQ40_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
7 REQ39_EN0	Writing a 1 to REQ39_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
6 REQ38_EN0	Writing a 1 to REQ38_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
5 REQ37_EN0	Writing a 1 to REQ37_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
4 REQ36_EN0	Writing a 1 to REQ36_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
3 REQ35_EN0	Writing a 1 to REQ35_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 REQ34_EN0	Writing a 1 to REQ34_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
1 REQ33_EN0	Writing a 1 to REQ33_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.
0 REQ32_EN0	Writing a 1 to REQ32_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE1.

26.5.1.62 DMA0 Request Enable1 (DMA0_REQ_ENABLE1_CLR)

Offset

Register	Offset
DMA0_REQ_ENABLE1_CLR	718h

Function

Writing a 1 to a bit in this register clears the corresponding bit in DMA0_REQ_ENABLE1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								0	0							
W	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...			REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ63_EN0	Writing a 1 to REQ63_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
30 REQ62_EN0	Writing a 1 to REQ62_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
29 REQ61_EN0	Writing a 1 to REQ61_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
28 REQ60_EN0	Writing a 1 to REQ60_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
27 REQ59_EN0	Writing a 1 to REQ59_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
26 REQ58_EN0	Writing a 1 to REQ58_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
25 REQ57_EN0	Writing a 1 to REQ57_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
24 —	Reserved
23 —	Reserved
22 REQ54_EN0	Writing a 1 to REQ54_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
21 REQ53_EN0	Writing a 1 to REQ53_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
20 REQ52_EN0	Writing a 1 to REQ52_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
19 REQ51_EN0	Writing a 1 to REQ51_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
18 REQ50_EN0	Writing a 1 to REQ50_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
17 REQ49_EN0	Writing a 1 to REQ49_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
16	Writing a 1 to REQ48_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ48_EN0	
15 REQ47_EN0	Writing a 1 to REQ47_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
14 REQ46_EN0	Writing a 1 to REQ46_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
13 REQ45_EN0	Writing a 1 to REQ45_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
12 REQ44_EN0	Writing a 1 to REQ44_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
11 REQ43_EN0	Writing a 1 to REQ43_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
10 REQ42_EN0	Writing a 1 to REQ42_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
9 REQ41_EN0	Writing a 1 to REQ41_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
8 REQ40_EN0	Writing a 1 to REQ40_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
7 REQ39_EN0	Writing a 1 to REQ39_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
6 REQ38_EN0	Writing a 1 to REQ38_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
5 REQ37_EN0	Writing a 1 to REQ37_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
4 REQ36_EN0	Writing a 1 to REQ36_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
3 REQ35_EN0	Writing a 1 to REQ35_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
2 REQ34_EN0	Writing a 1 to REQ34_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 REQ33_EN0	Writing a 1 to REQ33_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.
0 REQ32_EN0	Writing a 1 to REQ32_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE1.

26.5.1.63 DMA0 Request Enable1 (DMA0_REQ_ENABLE1_TOG)

Offset

Register	Offset
DMA0_REQ_ENABLE1_TOG	71Ch

Function

Writing a 1 to REQ_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								0	0							
W	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...			REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ63_EN0	Writing a 1 to REQ63_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
30 REQ62_EN0	Writing a 1 to REQ62_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 REQ61_EN0	Writing a 1 to REQ61_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
28 REQ60_EN0	Writing a 1 to REQ60_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
27 REQ59_EN0	Writing a 1 to REQ59_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
26 REQ58_EN0	Writing a 1 to REQ58_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
25 REQ57_EN0	Writing a 1 to REQ57_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
24 —	Reserved
23 —	Reserved
22 REQ54_EN0	Writing a 1 to REQ54_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
21 REQ53_EN0	Writing a 1 to REQ53_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
20 REQ52_EN0	Writing a 1 to REQ52_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
19 REQ51_EN0	Writing a 1 to REQ51_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
18 REQ50_EN0	Writing a 1 to REQ50_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
17 REQ49_EN0	Writing a 1 to REQ49_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
16 REQ48_EN0	Writing a 1 to REQ48_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
15	Writing a 1 to REQ47_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ47_EN0	
14 REQ46_EN0	Writing a 1 to REQ46_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
13 REQ45_EN0	Writing a 1 to REQ55_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
12 REQ44_EN0	Writing a 1 to REQ44_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
11 REQ43_EN0	Writing a 1 to REQ43_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
10 REQ42_EN0	Writing a 1 to REQ42_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
9 REQ41_EN0	Writing a 1 to REQ41_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
8 REQ40_EN0	Writing a 1 to REQ40_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
7 REQ39_EN0	Writing a 1 to REQ39_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
6 REQ38_EN0	Writing a 1 to REQ38_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
5 REQ37_EN0	Writing a 1 to REQ37_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
4 REQ36_EN0	Writing a 1 to REQ36_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
3 REQ35_EN0	Writing a 1 to REQ35_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
2 REQ34_EN0	Writing a 1 to REQ34_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.
1 REQ33_EN0	Writing a 1 to REQ33_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 REQ32_EN0	Writing a 1 to REQ32_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE1.

26.5.1.64 DMA0 Request Enable2 (DMA0_REQ_ENABLE2)

Offset

Register	Offset
DMA0_REQ_ENABLE2	720h

Function

DMA request 64-95 enable for DMA0. One bit per request. 0: DMA request to DMA0 and response from DMA0 are blocked. 1:DMA request and response are enabled for DMA0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...	0	0	REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 REQ95_EN0	This register is used to enable and disable I3C0 receive request. 0b - Disable 1b - Enable
30 REQ94_EN0	This register is used to enable and disable EMVSIM1 transmit request. 0b - Disable 1b - Enable
29 REQ93_EN0	This register is used to enable and disable EMVSIM1 receive request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
28 REQ92_EN0	This register is used to enable and disable EMVSIM0 transmit request. 0b - Disable 1b - Enable
27 REQ91_EN0	This register is used to enable and disable EMVSIM0 receive request. 0b - Disable 1b - Enable
26 —	Reserved
25 —	Reserved
24 REQ88_EN0	This register is used to enable and disable LP_FLEXCOMM9 transmit request. 0b - Disable 1b - Enable
23 REQ87_EN0	This register is used to enable and disable LP_FLEXCOMM9 receive request. 0b - Disable 1b - Enable
22 REQ86_EN0	This register is used to enable and disable LP_FLEXCOMM8 transmit request. 0b - Disable 1b - Enable
21 REQ85_EN0	This register is used to enable and disable LP_FLEXCOMM8 receive request. 0b - Disable 1b - Enable
20 REQ84_EN0	This register is used to enable and disable LP_FLEXCOMM7 transmit request. 0b - Disable 1b - Enable
19 REQ83_EN0	This register is used to enable and disable LP_FLEXCOMM7 receive request. 0b - Disable 1b - Enable
18	This register is used to enable and disable LP_FLEXCOMM6 transmit request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ82_EN0	0b - Disable 1b - Enable
17 REQ81_EN0	This register is used to enable and disable LP_FLEXCOMM6 receive request. 0b - Disable 1b - Enable
16 REQ80_EN0	This register is used to enable and disable LP_FLEXCOMM5 transmit request. 0b - Disable 1b - Enable
15 REQ79_EN0	This register is used to enable and disable LP_FLEXCOMM5 receive request. 0b - Disable 1b - Enable
14 REQ78_EN0	This register is used to enable and disable LP_FLEXCOMM4 transmit request. 0b - Disable 1b - Enable
13 REQ77_EN0	This register is used to enable and disable LP_FLEXCOMM4 receive request. 0b - Disable 1b - Enable
12 REQ76_EN0	This register is used to enable and disable LP_FLEXCOMM3 transmit request. 0b - Disable 1b - Enable
11 REQ75_EN0	This register is used to enable and disable LP_FLEXCOMM3 receive request. 0b - Disable 1b - Enable
10 REQ74_EN0	This register is used to enable and disable LP_FLEXCOMM2 transmit request. 0b - Disable 1b - Enable
9 REQ73_EN0	This register is used to enable and disable LP_FLEXCOMM2 receive request. 0b - Disable 1b - Enable
8 REQ72_EN0	This register is used to enable and disable LP_FLEXCOMM1 transmit request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ71_EN0	This register is used to enable and disable LP_FLEXCOMM1 receive request. 0b - Disable 1b - Enable
6 REQ70_EN0	This register is used to enable and disable LP_FLEXCOMM0 transmit request. 0b - Disable 1b - Enable
5 REQ69_EN0	This register is used to enable and disable LP_FLEXCOMM0 receive request. 0b - Disable 1b - Enable
4 REQ68_EN0	This register is used to enable and disable FlexIO0 shift register 7 request. 0b - Disable 1b - Enable
3 REQ67_EN0	This register is used to enable and disable FlexIO0 shift register 6 request. 0b - Disable 1b - Enable
2 REQ66_EN0	This register is used to enable and disable FlexIO0 shift register 5 request. 0b - Disable 1b - Enable
1 REQ65_EN0	This register is used to enable and disable FlexIO0 shift register 4 request. 0b - Disable 1b - Enable
0 REQ64_EN0	This register is used to enable and disable FlexIO0 shift register 3 request. 0b - Disable 1b - Enable

26.5.1.65 DMA0 Request Enable2 (DMA0_REQ_ENABLE2_SET)

Offset

Register	Offset
DMA0_REQ_ENABLE2_SET	724h

Function

Writing a 1 to a bit in this register sets the corresponding bit in DMA0_REQ_ENABLE2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R						0	0									
W	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...			REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ95_EN0	Writing a 1 to REQ95_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
30 REQ94_EN0	Writing a 1 to REQ94_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
29 REQ93_EN0	Writing a 1 to REQ93_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
28 REQ92_EN0	Writing a 1 to REQ92_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
27 REQ91_EN0	Writing a 1 to REQ91_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
26 —	Reserved
25 —	Reserved
24 REQ88_EN0	Writing a 1 to REQ88_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
23 REQ87_EN0	Writing a 1 to REQ87_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
22	Writing a 1 to REQ86_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ86_EN0	
21 REQ85_EN0	Writing a 1 to REQ85_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
20 REQ84_EN0	Writing a 1 to REQ84_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
19 REQ83_EN0	Writing a 1 to REQ83_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
18 REQ82_EN0	Writing a 1 to REQ82_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
17 REQ81_EN0	Writing a 1 to REQ81_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
16 REQ80_EN0	Writing a 1 to REQ80_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
15 REQ79_EN0	Writing a 1 to REQ79_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
14 REQ78_EN0	Writing a 1 to REQ78_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
13 REQ77_EN0	Writing a 1 to REQ77_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
12 REQ76_EN0	Writing a 1 to REQ876_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
11 REQ75_EN0	Writing a 1 to REQ75_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
10 REQ74_EN0	Writing a 1 to REQ74_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
9 REQ73_EN0	Writing a 1 to REQ73_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
8 REQ72_EN0	Writing a 1 to REQ72_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ71_EN0	Writing a 1 to REQ71_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
6 REQ70_EN0	Writing a 1 to REQ70_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
5 REQ69_EN0	Writing a 1 to REQ69_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
4 REQ68_EN0	Writing a 1 to REQ68_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
3 REQ67_EN0	Writing a 1 to REQ67_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
2 REQ66_EN0	Writing a 1 to REQ66_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
1 REQ65_EN0	Writing a 1 to REQ65_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.
0 REQ64_EN0	Writing a 1 to REQ64_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE2.

26.5.1.66 DMA0 Request Enable2 (DMA0_REQ_ENABLE2_CLR)

Offset

Register	Offset
DMA0_REQ_ENABLE2_CLR	728h

Function

Writing a 1 to a bit in this register clears the corresponding bit in DMA0_REQ_ENABLE2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R						0	0									
W	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...			REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ95_EN0	Writing a 1 to REQ95_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
30 REQ94_EN0	Writing a 1 to REQ94_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
29 REQ93_EN0	Writing a 1 to REQ93_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
28 REQ92_EN0	Writing a 1 to REQ92_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
27 REQ91_EN0	Writing a 1 to REQ91_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
26 —	Reserved
25 —	Reserved
24 REQ88_EN0	Writing a 1 to REQ88_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
23 REQ87_EN0	Writing a 1 to REQ87_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
22	Writing a 1 to REQ86_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ86_EN0	
21 REQ85_EN0	Writing a 1 to REQ85_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
20 REQ84_EN0	Writing a 1 to REQ84_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
19 REQ83_EN0	Writing a 1 to REQ83_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
18 REQ82_EN0	Writing a 1 to REQ82_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
17 REQ81_EN0	Writing a 1 to REQ81_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
16 REQ80_EN0	Writing a 1 to REQ80_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
15 REQ79_EN0	Writing a 1 to REQ79_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
14 REQ78_EN0	Writing a 1 to REQ78_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
13 REQ77_EN0	Writing a 1 to REQ77_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
12 REQ76_EN0	Writing a 1 to REQ76_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
11 REQ75_EN0	Writing a 1 to REQ75_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
10 REQ74_EN0	Writing a 1 to REQ74_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
9 REQ73_EN0	Writing a 1 to REQ73_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
8 REQ72_EN0	Writing a 1 to REQ72_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ71_EN0	Writing a 1 to REQ71_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
6 REQ70_EN0	Writing a 1 to REQ70_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
5 REQ69_EN0	Writing a 1 to REQ69_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
4 REQ68_EN0	Writing a 1 to REQ68_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
3 REQ67_EN0	Writing a 1 to REQ67_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
2 REQ66_EN0	Writing a 1 to REQ66_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
1 REQ65_EN0	Writing a 1 to REQ65_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.
0 REQ64_EN0	Writing a 1 to REQ64_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE2.

26.5.1.67 DMA0 Request Enable2 (DMA0_REQ_ENABLE2_TOG)

Offset

Register	Offset
DMA0_REQ_ENABLE2_TOG	72Ch

Function

Writing a 1 to a bit in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R						0	0									
W	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...			REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ95_EN0	Writing a 1 to REQ95_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
30 REQ94_EN0	Writing a 1 to REQ94_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
29 REQ93_EN0	Writing a 1 to REQ93_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
28 REQ92_EN0	Writing a 1 to REQ92_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
27 REQ91_EN0	Writing a 1 to REQ91_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
26 —	Reserved
25 —	Reserved
24 REQ88_EN0	Writing a 1 to REQ88_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
23 REQ87_EN0	Writing a 1 to REQ87_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
22	Writing a 1 to REQ86_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ86_EN0	
21 REQ85_EN0	Writing a 1 to REQ85_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
20 REQ84_EN0	Writing a 1 to REQ84_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
19 REQ83_EN0	Writing a 1 to REQ83_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
18 REQ82_EN0	Writing a 1 to REQ82_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
17 REQ81_EN0	Writing a 1 to REQ81_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
16 REQ80_EN0	Writing a 1 to REQ80_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
15 REQ79_EN0	Writing a 1 to REQ79_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
14 REQ78_EN0	Writing a 1 to REQ78_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
13 REQ77_EN0	Writing a 1 to REQ77_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
12 REQ76_EN0	Writing a 1 to REQ76_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
11 REQ75_EN0	Writing a 1 to REQ75_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
10 REQ74_EN0	Writing a 1 to REQ74_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
9 REQ73_EN0	Writing a 1 to REQ73_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
8 REQ72_EN0	Writing a 1 to REQ72_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ71_EN0	Writing a 1 to REQ71_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
6 REQ70_EN0	Writing a 1 to REQ70_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
5 REQ69_EN0	Writing a 1 to REQ69_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
4 REQ68_EN0	Writing a 1 to REQ68_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
3 REQ67_EN0	Writing a 1 to REQ67_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
2 REQ66_EN0	Writing a 1 to REQ66_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
1 REQ65_EN0	Writing a 1 to REQ65_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.
0 REQ64_EN0	Writing a 1 to REQ64_EN0 in this register toggles the corresponding bit in DMA0_REQ_ENABLE2.

26.5.1.68 DMA0 Request Enable3 (DMA0_REQ_ENABLE3)

Offset

Register	Offset
DMA0_REQ_ENABLE3	730h

Function

DMA request 96-121 enable for DMA0. One bit per request. 0: DMA request to DMA0 and response from DMA0 are blocked. 1:DMA request and response are enabled for DMA0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	REQ121_...	REQ120_...	REQ119_...	REQ118_...	REQ117_...	REQ116_...	REQ115_...	REQ114_...	REQ113_...	REQ112_...
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REQ111_...	REQ110_...	REQ109_...	REQ108_...	REQ107_...	REQ106_...	REQ105_...	REQ104_...	REQ103_...	REQ102_...	REQ101_...	REQ100_...	REQ99_E...	REQ98_E...	REQ97_E...	REQ96_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 REQ121_EN0	This register is used to enable and disable TSI0 out of range request. 0b - Disable 1b - Enable
24 REQ120_EN0	This register is used to enable and disable TSI0 end of scan request. 0b - Disable 1b - Enable
23 REQ119_EN0	This register is used to enable and disable GPIO5 pin event request 1. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 REQ118_EN0	This register is used to enable and disable GPIO5 pin event request 0. 0b - Disable 1b - Enable
21 REQ117_EN0	This register is used to enable and disable GPIO4 pin event request 1. 0b - Disable 1b - Enable
20 REQ116_EN0	This register is used to enable and disable GPIO4 pin event request 0. 0b - Disable 1b - Enable
19 REQ115_EN0	This register is used to enable and disable GPIO3 pin event request 1. 0b - Disable 1b - Enable
18 REQ114_EN0	This register is used to enable and disable GPIO3 pin event request 0. 0b - Disable 1b - Enable
17 REQ113_EN0	This register is used to enable and disable GPIO2 pin event request 1. 0b - Disable 1b - Enable
16 REQ112_EN0	This register is used to enable and disable GPIO2 pin event request 0. 0b - Disable 1b - Enable
15 REQ111_EN0	This register is used to enable and disable GPIO1 pin event request 1. 0b - Disable 1b - Enable
14 REQ110_EN0	This register is used to enable and disable GPIO1 pin event request 0. 0b - Disable 1b - Enable
13 REQ109_EN0	This register is used to enable and disable GPIO0 pin event request 1. 0b - Disable 1b - Enable
12 REQ108_EN0	This register is used to enable and disable GPIO0 pin event request 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
11 REQ107_EN0	This register is used to enable and disable SINC0 ipd_req_sinc[4] or ipd_req_alt [4] request. 0b - Disable 1b - Enable
10 REQ106_EN0	This register is used to enable and disable SINC0 ipd_req_sinc[3] or ipd_req_alt [3] request. 0b - Disable 1b - Enable
9 REQ105_EN0	This register is used to enable and disable SINC0 ipd_req_sinc[2] or ipd_req_alt [2] request. 0b - Disable 1b - Enable
8 REQ104_EN0	This register is used to enable and disable SINC0 ipd_req_sinc[1] or ipd_req_alt [1] request. 0b - Disable 1b - Enable
7 REQ103_EN0	This register is used to enable and disable SINC0 ipd_req_sinc[0] or ipd_req_alt [0] request. 0b - Disable 1b - Enable
6 REQ102_EN0	This register is used to enable and disable SAI1 transmit request. 0b - Disable 1b - Enable
5 REQ101_EN0	This register is used to enable and disable SAI1 receive request. 0b - Disable 1b - Enable
4 REQ100_EN0	This register is used to enable and disable SAI0 transmit request. 0b - Disable 1b - Enable
3 REQ99_EN0	This register is used to enable and disable SAI0 receive request. 0b - Disable 1b - Enable
2 REQ98_EN0	This register is used to enable and disable I3C1 transmit request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 REQ97_EN0	This register is used to enable and disable I3C1 receive request. 0b - Disable 1b - Enable
0 REQ96_EN0	This register is used to enable and disable I3C0 transmit request. 0b - Disable 1b - Enable

26.5.1.69 DMA0 Request Enable3 (DMA0_REQ_ENABLE3_SET)

Offset

Register	Offset
DMA0_REQ_ENABLE3_SET	734h

Function

Writing a 1 to a bit in this register sets the corresponding bit in DMA0_REQ_ENABLE3

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0										
W							REQ121_...	REQ120_...	REQ119_...	REQ118_...	REQ117_...	REQ116_...	REQ115_...	REQ114_...	REQ113_...	REQ112_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ111_...	REQ110_...	REQ109_...	REQ108_...	REQ107_...	REQ106_...	REQ105_...	REQ104_...	REQ103_...	REQ102_...	REQ101_...	REQ100_...	REQ99_E...	REQ98_E...	REQ97_E...	REQ96_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 REQ121_EN0	Writing a 1 to REQ121_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
24 REQ120_EN0	Writing a 1 to REQ120_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
23 REQ119_EN0	Writing a 1 to REQ119_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
22 REQ118_EN0	Writing a 1 to REQ118_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
21 REQ117_EN0	Writing a 1 to REQ117_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
20 REQ116_EN0	Writing a 1 to REQ116_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
19 REQ115_EN0	Writing a 1 to REQ115_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
18 REQ114_EN0	Writing a 1 to REQ114_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
17 REQ113_EN0	Writing a 1 to REQ113_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
16 REQ112_EN0	Writing a 1 to REQ112_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 REQ111_EN0	Writing a 1 to REQ111_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
14 REQ110_EN0	Writing a 1 to REQ110_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
13 REQ109_EN0	Writing a 1 to REQ109_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
12 REQ108_EN0	Writing a 1 to REQ108_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
11 REQ107_EN0	Writing a 1 to REQ107_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
10 REQ106_EN0	Writing a 1 to REQ106_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
9 REQ105_EN0	Writing a 1 to REQ105_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
8 REQ104_EN0	Writing a 1 to REQ104_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
7 REQ103_EN0	Writing a 1 to REQ103_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
6 REQ102_EN0	Writing a 1 to REQ102_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
5 REQ101_EN0	Writing a 1 to REQ101_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
4 REQ100_EN0	Writing a 1 to REQ100_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
3 REQ99_EN0	Writing a 1 to REQ99_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
2 REQ98_EN0	Writing a 1 to REQ98_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
1	Writing a 1 to REQ97_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ97_EN0	
0 REQ96_EN0	Writing a 1 to REQ96_EN0 in this register sets the corresponding bit in DMA0_REQ_ENABLE3

26.5.1.70 DMA0 Request Enable3 (DMA0_REQ_ENABLE3_CLR)

Offset

Register	Offset
DMA0_REQ_ENABLE3_CLR	738h

Function

Writing a 1 to a bit in this register clears the corresponding bit in DMA0_REQ_ENABLE3

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0										
W							REQ121_...	REQ120_...	REQ119_...	REQ118_...	REQ117_...	REQ116_...	REQ115_...	REQ114_...	REQ113_...	REQ112_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ111_...	REQ110_...	REQ109_...	REQ108_...	REQ107_...	REQ106_...	REQ105_...	REQ104_...	REQ103_...	REQ102_...	REQ101_...	REQ100_...	REQ99_E...	REQ98_E...	REQ97_E...	REQ96_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 REQ121_EN0	Writing a 1 to REQ121_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
24 REQ120_EN0	Writing a 1 to REQ120_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
23 REQ119_EN0	Writing a 1 to REQ119_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
22 REQ118_EN0	Writing a 1 to REQ118_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
21 REQ117_EN0	Writing a 1 to REQ117_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
20 REQ116_EN0	Writing a 1 to REQ116_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
19 REQ115_EN0	Writing a 1 to REQ115_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
18 REQ114_EN0	Writing a 1 to REQ114_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
17 REQ113_EN0	Writing a 1 to REQ113_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
16 REQ112_EN0	Writing a 1 to REQ112_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
15 REQ111_EN0	Writing a 1 to REQ111_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
14 REQ110_EN0	Writing a 1 to REQ110_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
13 REQ109_EN0	Writing a 1 to REQ109_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
12 REQ108_EN0	Writing a 1 to REQ108_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
11 REQ107_EN0	Writing a 1 to REQ107_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
10 REQ106_EN0	Writing a 1 to REQ106_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
9 REQ105_EN0	Writing a 1 to REQ105_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
8 REQ104_EN0	Writing a 1 to REQ104_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
7 REQ103_EN0	Writing a 1 to REQ103_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
6 REQ102_EN0	Writing a 1 to REQ102_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
5 REQ101_EN0	Writing a 1 to REQ101_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
4 REQ100_EN0	Writing a 1 to REQ100_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
3 REQ99_EN0	Writing a 1 to REQ99_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
2 REQ98_EN0	Writing a 1 to REQ98_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
1 REQ97_EN0	Writing a 1 to REQ97_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
0	Writing a 1 to REQ96_EN0 in this register clears the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ96_EN0	

26.5.1.71 DMA1 Request Enable0 (DMA1_REQ_ENABLE0)

Offset

Register	Offset
DMA1_REQ_ENABLE0	780h

Function

DMA request 0-31 enable for DMA1. One bit per request. 0: DMA request to DMA1 and response from DMA1 are blocked. 1:DMA request and response are enabled for DMA1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN1	REQ8_EN1	REQ7_EN1	REQ6_EN1	REQ5_EN1	REQ4_EN1	REQ3_EN1	REQ2_EN1	REQ1_EN1	0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 REQ31_EN1	This register is used to enable and disable EVTG0 OUT0A request. 0b - Disable 1b - Enable
30 REQ30_EN1	This register is used to enable and disable CMP2 DMA_request. 0b - Disable 1b - Enable
29 REQ29_EN1	This register is used to enable and disable CMP1 DMA_request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
28 REQ28_EN1	This register is used to enable and disable CMP0 DMA_request. 0b - Disable 1b - Enable
27 REQ27_EN1	This register is used to enable and disable DAC2 FIFO_request. 0b - Disable 1b - Enable
26 REQ26_EN1	This register is used to enable and disable DAC1 FIFO_request. 0b - Disable 1b - Enable
25 REQ25_EN1	This register is used to enable and disable DAC0 FIFO_request. 0b - Disable 1b - Enable
24 REQ24_EN1	This register is used to enable and disable ADC1 FIFO B request. 0b - Disable 1b - Enable
23 REQ23_EN1	This register is used to enable and disable ADC1 FIFO A request. 0b - Disable 1b - Enable
22 REQ22_EN1	This register is used to enable and disable ADC0 FIFO B request. 0b - Disable 1b - Enable
21 REQ21_EN1	This register is used to enable and disable ADC0 FIFO A request. 0b - Disable 1b - Enable
20 REQ20_EN1	This register is used to enable and disable SCT0 DMA1 request. 0b - Disable 1b - Enable
19 REQ19_EN1	This register is used to enable and disable SCT0 DMA0 request. 0b - Disable 1b - Enable
18 REQ18_EN1	This register is used to enable and disable MICFIL0 FIFO_request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
17 REQ17_EN1	This register is used to enable and disable WUU0 wake up event request. 0b - Disable 1b - Enable
16 REQ16_EN1	This register is used to enable and disable CTIMER4 DMAREQ_M1 request. 0b - Disable 1b - Enable
15 REQ15_EN1	This register is used to enable and disable CTIMER4 DMAREQ_M0 request. 0b - Disable 1b - Enable
14 REQ14_EN1	This register is used to enable and disable CTIMER3 DMAREQ_M1 request. 0b - Disable 1b - Enable
13 REQ13_EN1	This register is used to enable and disable CTIMER3 DMAREQ_M0 request. 0b - Disable 1b - Enable
12 REQ12_EN1	This register is used to enable and disable CTIMER2 DMAREQ_M1 request. 0b - Disable 1b - Enable
11 REQ11_EN1	This register is used to enable and disable CTIMER2 DMAREQ_M0 request. 0b - Disable 1b - Enable
10 REQ10_EN1	This register is used to enable and disable CTIMER1 DMAREQ_M1 request. 0b - Disable 1b - Enable
9 REQ9_EN1	This register is used to enable and disable CTIMER1 DMAREQ_M0 request. 0b - Disable 1b - Enable
8 REQ8_EN1	This register is used to enable and disable CTIMER0 DMAREQ_M1 request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ7_EN1	This register is used to enable and disable CTIMER0 DMAREQ_M0 request. 0b - Disable 1b - Enable
6 REQ6_EN1	This register is used to enable and disable PINT0 INT3 request. 0b - Disable 1b - Enable
5 REQ5_EN1	This register is used to enable and disable PINT0 INT2 request. 0b - Disable 1b - Enable
4 REQ4_EN1	This register is used to enable and disable PINT0 INT1 request. 0b - Disable 1b - Enable
3 REQ3_EN1	This register is used to enable and disable PINT0 INT0 request. 0b - Disable 1b - Enable
2 REQ2_EN1	This register is used to enable and disable FLEXSPI0 transmit event request. 0b - Disable 1b - Enable
1 REQ1_EN1	This register is used to enable and disable FLEXSPI0 receive event request. 0b - Disable 1b - Enable
0 —	Reserved

26.5.1.72 DMA1 Request Enable0 (DMA1_REQ_ENABLE0_SET)

Offset

Register	Offset
DMA1_REQ_ENABLE0_SET	784h

Function
Writing a 1 to a bit in this register sets the corresponding bit in DMA1_REQ_ENABLE0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																0
W	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN1	REQ8_EN1	REQ7_EN1	REQ6_EN1	REQ5_EN1	REQ4_EN1	REQ3_EN1	REQ2_EN1	REQ1_EN1	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ31_EN1	Writing a 1 to REQ31_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
30 REQ30_EN1	Writing a 1 to REQ30_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
29 REQ29_EN1	Writing a 1 to REQ29_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
28 REQ28_EN1	Writing a 1 to REQ28_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
27 REQ27_EN1	Writing a 1 to REQ27_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
26 REQ26_EN1	Writing a 1 to REQ26_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
25 REQ25_EN1	Writing a 1 to REQ25_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
24 REQ24_EN1	Writing a 1 to REQ24_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
23 REQ23_EN1	Writing a 1 to REQ23_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
22	Writing a 1 to REQ22_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ22_EN1	
21 REQ21_EN1	Writing a 1 to REQ21_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
20 REQ20_EN1	Writing a 1 to REQ20_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
19 REQ19_EN1	Writing a 1 to REQ19_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
18 REQ18_EN1	Writing a 1 to REQ18_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
17 REQ17_EN1	Writing a 1 to REQ17_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
16 REQ16_EN1	Writing a 1 to REQ16_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
15 REQ15_EN1	Writing a 1 to REQ15_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
14 REQ14_EN1	Writing a 1 to REQ14_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
13 REQ13_EN1	Writing a 1 to REQ13_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
12 REQ12_EN1	Writing a 1 to REQ12_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
11 REQ11_EN1	Writing a 1 to REQ11_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
10 REQ10_EN1	Writing a 1 to REQ10_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
9 REQ9_EN1	Writing a 1 to REQ9_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
8 REQ8_EN1	Writing a 1 to REQ8_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ7_EN1	Writing a 1 to REQ7_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
6 REQ6_EN1	Writing a 1 to REQ6_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
5 REQ5_EN1	Writing a 1 to REQ5_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
4 REQ4_EN1	Writing a 1 to REQ4_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
3 REQ3_EN1	Writing a 1 to REQ3_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
2 REQ2_EN1	Writing a 1 to REQ2_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
1 REQ1_EN1	Writing a 1 to REQ1_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE0.
0 —	Reserved

26.5.1.73 DMA1 Request Enable0 (DMA1_REQ_ENABLE0_CLR)

Offset

Register	Offset
DMA1_REQ_ENABLE0_CLR	788h

Function

Writing a 1 to a bit in this register clears the corresponding bit in DMA1_REQ_ENABLE0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																0
W	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN1	REQ8_EN1	REQ7_EN1	REQ6_EN1	REQ5_EN1	REQ4_EN1	REQ3_EN1	REQ2_EN1	REQ1_EN1	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ31_EN1	Writing a 1 to REQ31_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
30 REQ30_EN1	Writing a 1 to REQ30_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
29 REQ29_EN1	Writing a 1 to REQ29_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
28 REQ28_EN1	Writing a 1 to REQ28_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
27 REQ27_EN1	Writing a 1 to REQ27_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
26 REQ26_EN1	Writing a 1 to REQ26_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
25 REQ25_EN1	Writing a 1 to REQ25_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
24 REQ24_EN1	Writing a 1 to REQ24_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
23 REQ23_EN1	Writing a 1 to REQ23_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
22	Writing a 1 to REQ22_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ22_EN1	
21 REQ21_EN1	Writing a 1 to REQ21_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
20 REQ20_EN1	Writing a 1 to REQ20_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
19 REQ19_EN1	Writing a 1 to REQ19_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
18 REQ18_EN1	Writing a 1 to REQ18_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
17 REQ17_EN1	Writing a 1 to REQ17_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
16 REQ16_EN1	Writing a 1 to REQ16_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
15 REQ15_EN1	Writing a 1 to REQ15_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
14 REQ14_EN1	Writing a 1 to REQ14_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
13 REQ13_EN1	Writing a 1 to REQ13_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
12 REQ12_EN1	Writing a 1 to REQ12_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
11 REQ11_EN1	Writing a 1 to REQ11_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
10 REQ10_EN1	Writing a 1 to REQ10_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
9 REQ9_EN1	Writing a 1 to REQ9_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
8 REQ8_EN1	Writing a 1 to REQ8_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ7_EN1	Writing a 1 to REQ7_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
6 REQ6_EN1	Writing a 1 to REQ6_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
5 REQ5_EN1	Writing a 1 to REQ5_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
4 REQ4_EN1	Writing a 1 to REQ4_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
3 REQ3_EN1	Writing a 1 to REQ3_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
2 REQ2_EN1	Writing a 1 to REQ2_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
1 REQ1_EN1	Writing a 1 to REQ1_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE0.
0 —	Reserved

26.5.1.74 DMA1 Request Enable0 (DMA1_REQ_ENABLE0_TOG)

Offset

Register	Offset
DMA1_REQ_ENABLE0_TOG	78Ch

Function

Writing a 1 to a bit in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ31_E...	REQ30_E...	REQ29_E...	REQ28_E...	REQ27_E...	REQ26_E...	REQ25_E...	REQ24_E...	REQ23_E...	REQ22_E...	REQ21_E...	REQ20_E...	REQ19_E...	REQ18_E...	REQ17_E...	REQ16_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																0
W	REQ15_E...	REQ14_E...	REQ13_E...	REQ12_E...	REQ11_E...	REQ10_E...	REQ9_EN1	REQ8_EN1	REQ7_EN1	REQ6_EN1	REQ5_EN1	REQ4_EN1	REQ3_EN1	REQ2_EN1	REQ1_EN1	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ31_EN1	Writing a 1 to REQ31_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
30 REQ30_EN1	Writing a 1 to REQ30_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
29 REQ29_EN1	Writing a 1 to REQ29_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
28 REQ28_EN1	Writing a 1 to REQ28_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
27 REQ27_EN1	Writing a 1 to REQ27_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
26 REQ26_EN1	Writing a 1 to REQ26_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
25 REQ25_EN1	Writing a 1 to REQ25_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
24 REQ24_EN1	Writing a 1 to REQ24_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
23 REQ23_EN1	Writing a 1 to REQ23_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
22	Writing a 1 to REQ22_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ22_EN1	
21 REQ21_EN1	Writing a 1 to REQ21_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
20 REQ20_EN1	Writing a 1 to REQ20_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
19 REQ19_EN1	Writing a 1 to REQ19_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
18 REQ18_EN1	Writing a 1 to REQ18_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
17 REQ17_EN1	Writing a 1 to REQ17_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
16 REQ16_EN1	Writing a 1 to REQ16_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
15 REQ15_EN1	Writing a 1 to REQ15_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
14 REQ14_EN1	Writing a 1 to REQ14_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
13 REQ13_EN1	Writing a 1 to REQ13_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
12 REQ12_EN1	Writing a 1 to REQ12_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
11 REQ11_EN1	Writing a 1 to REQ11_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
10 REQ10_EN1	Writing a 1 to REQ10_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
9 REQ9_EN1	Writing a 1 to RE9_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
8 REQ8_EN1	Writing a 1 to REQ8_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ7_EN1	Writing a 1 to REQ7_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
6 REQ6_EN1	Writing a 1 to REQ6_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
5 REQ5_EN1	Writing a 1 to REQ5_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
4 REQ4_EN1	Writing a 1 to REQ4_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
3 REQ3_EN1	Writing a 1 to REQ3_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
2 REQ2_EN1	Writing a 1 to REQ2_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
1 REQ1_EN1	Writing a 1 to REQ1_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE0.
0 —	Reserved

26.5.1.75 DMA1 Request Enable1 (DMA1_REQ_ENABLE1)

Offset

Register	Offset
DMA1_REQ_ENABLE1	790h

Function

DMA request 32-63 enable for DMA1. One bit per request. 0: DMA request to DMA1 and response from DMA1 are blocked. 1:DMA request and response are enabled for DMA1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...	0	0	REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 REQ63_EN1	This register is used to enable and disable FlexIO0 Shifter2 Status DMA request OR Timer2 Status DMA request. 0b - Disable 1b - Enable
30 REQ62_EN1	This register is used to enable and disable FlexIO0 Shifter1 Status DMA request OR Timer1 Status DMA request. 0b - Disable 1b - Enable
29 REQ61_EN1	This register is used to enable and disable FlexIO0 Shifter0 Status DMA request OR Timer0 Status DMA request. 0b - Disable 1b - Enable
28 REQ60_EN1	This register is used to enable and disable CAN1 DMA request. 0b - Disable 1b - Enable
27 REQ59_EN1	This register is used to enable and disable CAN0 DMA request. 0b - Disable 1b - Enable
26 REQ58_EN1	This register is used to enable and disable LPTMR1 counter match event request. 0b - Disable 1b - Enable
25 REQ57_EN1	This register is used to enable and disable LPTMR0 counter match event request. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
24 —	Reserved
23 —	Reserved
22 REQ54_EN1	This register is used to enable and disable PWM1 Req_val3 request. 0b - Disable 1b - Enable
21 REQ53_EN1	This register is used to enable and disable PWM1 Req_val2 request. 0b - Disable 1b - Enable
20 REQ52_EN1	This register is used to enable and disable PWM1 Req_val1 request. 0b - Disable 1b - Enable
19 REQ51_EN1	This register is used to enable and disable PWM1 Req_val0 request. 0b - Disable 1b - Enable
18 REQ50_EN1	This register is used to enable and disable PWM1 Req_capt3 request. 0b - Disable 1b - Enable
17 REQ49_EN1	This register is used to enable and disable PWM1 Req_capt2 request. 0b - Disable 1b - Enable
16 REQ48_EN1	This register is used to enable and disable PWM1 Req_capt1 request. 0b - Disable 1b - Enable
15 REQ47_EN1	This register is used to enable and disable PWM1 Req_capt0 request. 0b - Disable 1b - Enable
14 REQ46_EN1	This register is used to enable and disable PWM0 Req_val3 request. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
13 REQ45_EN1	This register is used to enable and disable PWM0 Req_val2 request. 0b - Disable 1b - Enable
12 REQ44_EN1	This register is used to enable and disable PWM0 Req_val1 request. 0b - Disable 1b - Enable
11 REQ43_EN1	This register is used to enable and disable PWM0 Req_val0 request. 0b - Disable 1b - Enable
10 REQ42_EN1	This register is used to enable and disable PWM0 Req_capt3 request. 0b - Disable 1b - Enable
9 REQ41_EN1	This register is used to enable and disable PWM0 Req_capt2 request. 0b - Disable 1b - Enable
8 REQ40_EN1	This register is used to enable and disable PWM0 Req_capt1 request. 0b - Disable 1b - Enable
7 REQ39_EN1	This register is used to enable and disable PWM0 Req_capt0 request. 0b - Disable 1b - Enable
6 REQ38_EN1	This register is used to enable and disable EVTG0 OUT3B request. 0b - Disable 1b - Enable
5 REQ37_EN1	This register is used to enable and disable EVTG0 OUT3A request. 0b - Disable 1b - Enable
4 REQ36_EN1	This register is used to enable and disable EVTG0 OUT2B request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 REQ35_EN1	This register is used to enable and disable EVTG0 OUT2A request. 0b - Disable 1b - Enable
2 REQ34_EN1	This register is used to enable and disable EVTG0 OUT1B request. 0b - Disable 1b - Enable
1 REQ33_EN1	This register is used to enable and disable EVTG0 OUT1A request. 0b - Disable 1b - Enable
0 REQ32_EN1	This register is used to enable and disable EVTG0 OUT0B request. 0b - Disable 1b - Enable

26.5.1.76 DMA1 Request Enable1 (DMA1_REQ_ENABLE1_SET)

Offset

Register	Offset
DMA1_REQ_ENABLE1_SET	794h

Function

Writing a 1 to REQ_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								0	0							
W	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...			REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ63_EN1	Writing a 1 to REQ63_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
30 REQ62_EN1	Writing a 1 to REQ62_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
29 REQ61_EN1	Writing a 1 to REQ61_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
28 REQ60_EN1	Writing a 1 to REQ60_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
27 REQ59_EN1	Writing a 1 to REQ59_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
26 REQ58_EN1	Writing a 1 to REQ58_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
25 REQ57_EN1	Writing a 1 to REQ57_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
24 —	Reserved
23 —	Reserved
22 REQ54_EN1	Writing a 1 to REQ54_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
21 REQ53_EN1	Writing a 1 to REQ53_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
20 REQ52_EN1	Writing a 1 to REQ52_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
19 REQ51_EN1	Writing a 1 to REQ51_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
18 REQ50_EN1	Writing a 1 to REQ50_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
17	Writing a 1 to REQ49_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ49_EN1	
16 REQ48_EN1	Writing a 1 to REQ48_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
15 REQ47_EN1	Writing a 1 to REQ47_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
14 REQ46_EN1	Writing a 1 to REQ46_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
13 REQ45_EN1	Writing a 1 to REQ45_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
12 REQ44_EN1	Writing a 1 to REQ44_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
11 REQ43_EN1	Writing a 1 to REQ43_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
10 REQ42_EN1	Writing a 1 to REQ42_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
9 REQ41_EN1	Writing a 1 to REQ41_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
8 REQ40_EN1	Writing a 1 to REQ40_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
7 REQ39_EN1	Writing a 1 to REQ39_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
6 REQ38_EN1	Writing a 1 to REQ38_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
5 REQ37_EN1	Writing a 1 to REQ37_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
4 REQ36_EN1	Writing a 1 to REQ36_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
3 REQ35_EN1	Writing a 1 to REQ35_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 REQ34_EN1	Writing a 1 to REQ34_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
1 REQ33_EN1	Writing a 1 to REQ33_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.
0 REQ32_EN1	Writing a 1 to REQ32_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE1.

26.5.1.77 DMA1 Request Enable1 (DMA1_REQ_ENABLE1_CLR)

Offset

Register	Offset
DMA1_REQ_ENABLE1_CLR	798h

Function

Writing a 1 to a bit in this register clears the corresponding bit in DMA1_REQ_ENABLE1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								0	0							
W	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...			REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ63_EN1	Writing a 1 to REQ63_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
30 REQ62_EN1	Writing a 1 to REQ62_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
29 REQ61_EN1	Writing a 1 to REQ61_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
28 REQ60_EN1	Writing a 1 to REQ60_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
27 REQ59_EN1	Writing a 1 to REQ59_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
26 REQ58_EN1	Writing a 1 to REQ58_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
25 REQ57_EN1	Writing a 1 to REQ57_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
24 —	Reserved
23 —	Reserved
22 REQ54_EN1	Writing a 1 to REQ54_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
21 REQ53_EN1	Writing a 1 to REQ53_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
20 REQ52_EN1	Writing a 1 to REQ52_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
19 REQ51_EN1	Writing a 1 to REQ51_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
18 REQ50_EN1	Writing a 1 to REQ50_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
17 REQ49_EN1	Writing a 1 to REQ49_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
16	Writing a 1 to REQ48_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ48_EN1	
15 REQ47_EN1	Writing a 1 to REQ47_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
14 REQ46_EN1	Writing a 1 to REQ46_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
13 REQ45_EN1	Writing a 1 to REQ45_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
12 REQ44_EN1	Writing a 1 to REQ44_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
11 REQ43_EN1	Writing a 1 to REQ43_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
10 REQ42_EN1	Writing a 1 to REQ42_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
9 REQ41_EN1	Writing a 1 to REQ41_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
8 REQ40_EN1	Writing a 1 to REQ40_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
7 REQ39_EN1	Writing a 1 to REQ39_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
6 REQ38_EN1	Writing a 1 to REQ38_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
5 REQ37_EN1	Writing a 1 to REQ37_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
4 REQ36_EN1	Writing a 1 to REQ36_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
3 REQ35_EN1	Writing a 1 to REQ35_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
2 REQ34_EN1	Writing a 1 to REQ34_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 REQ33_EN1	Writing a 1 to REQ33_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.
0 REQ32_EN1	Writing a 1 to REQ32_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE1.

26.5.1.78 DMA1 Request Enable1 (DMA1_REQ_ENABLE1_TOG)

Offset

Register	Offset
DMA1_REQ_ENABLE1_TOG	79Ch

Function

Writing a 1 to REQ_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								0	0							
W	REQ63_E...	REQ62_E...	REQ61_E...	REQ60_E...	REQ59_E...	REQ58_E...	REQ57_E...			REQ54_E...	REQ53_E...	REQ52_E...	REQ51_E...	REQ50_E...	REQ49_E...	REQ48_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ47_E...	REQ46_E...	REQ45_E...	REQ44_E...	REQ43_E...	REQ42_E...	REQ41_E...	REQ40_E...	REQ39_E...	REQ38_E...	REQ37_E...	REQ36_E...	REQ35_E...	REQ34_E...	REQ33_E...	REQ32_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ63_EN1	Writing a 1 to REQ63_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
30 REQ62_EN1	Writing a 1 to REQ62_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 REQ61_EN1	Writing a 1 to REQ61_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
28 REQ60_EN1	Writing a 1 to REQ60_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
27 REQ59_EN1	Writing a 1 to REQ59_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
26 REQ58_EN1	Writing a 1 to REQ58_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
25 REQ57_EN1	Writing a 1 to REQ57_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
24 —	Reserved
23 —	Reserved
22 REQ54_EN1	Writing a 1 to REQ54_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
21 REQ53_EN1	Writing a 1 to REQ53_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
20 REQ52_EN1	Writing a 1 to REQ52_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
19 REQ51_EN1	Writing a 1 to REQ51_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
18 REQ50_EN1	Writing a 1 to REQ50_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
17 REQ49_EN1	Writing a 1 to REQ49_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
16 REQ48_EN1	Writing a 1 to REQ48_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
15	Writing a 1 to REQ47_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ47_EN1	
14 REQ46_EN1	Writing a 1 to REQ46_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
13 REQ45_EN1	Writing a 1 to REQ55_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
12 REQ44_EN1	Writing a 1 to REQ44_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
11 REQ43_EN1	Writing a 1 to REQ43_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
10 REQ42_EN1	Writing a 1 to REQ42_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
9 REQ41_EN1	Writing a 1 to REQ41_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
8 REQ40_EN1	Writing a 1 to REQ40_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
7 REQ39_EN1	Writing a 1 to REQ39_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
6 REQ38_EN1	Writing a 1 to REQ38_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
5 REQ37_EN1	Writing a 1 to REQ37_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
4 REQ36_EN1	Writing a 1 to REQ36_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
3 REQ35_EN1	Writing a 1 to REQ35_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
2 REQ34_EN1	Writing a 1 to REQ34_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.
1 REQ33_EN1	Writing a 1 to REQ33_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 REQ32_EN1	Writing a 1 to REQ32_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE1.

26.5.1.79 DMA1 Request Enable2 (DMA1_REQ_ENABLE2)

Offset

Register	Offset
DMA1_REQ_ENABLE2	7A0h

Function

DMA request 64-95 enable for DMA1. One bit per request. 0: DMA request to DMA1 and response from DMA1 are blocked. 1:DMA request and response are enabled for DMA1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...	0	0	REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 REQ95_EN1	This register is used to enable and disable I3C0 receive request. 0b - Disable 1b - Enable
30 REQ94_EN1	This register is used to enable and disable EMVSIM1 transmit request. 0b - Disable 1b - Enable
29 REQ93_EN1	This register is used to enable and disable EMVSIM1 receive request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
28 REQ92_EN1	This register is used to enable and disable EMVSIM0 transmit request. 0b - Disable 1b - Enable
27 REQ91_EN1	This register is used to enable and disable EMVSIM0 receive request. 0b - Disable 1b - Enable
26 —	Reserved
25 —	Reserved
24 REQ88_EN1	This register is used to enable and disable LP_FLEXCOMM9 transmit request. 0b - Disable 1b - Enable
23 REQ87_EN1	This register is used to enable and disable LP_FLEXCOMM9 receive request. 0b - Disable 1b - Enable
22 REQ86_EN1	This register is used to enable and disable LP_FLEXCOMM8 transmit request. 0b - Disable 1b - Enable
21 REQ85_EN1	This register is used to enable and disable LP_FLEXCOMM8 receive request. 0b - Disable 1b - Enable
20 REQ84_EN1	This register is used to enable and disable LP_FLEXCOMM7 transmit request. 0b - Disable 1b - Enable
19 REQ83_EN1	This register is used to enable and disable LP_FLEXCOMM7 receive request. 0b - Disable 1b - Enable
18	This register is used to enable and disable LP_FLEXCOMM6 transmit request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ82_EN1	0b - Disable 1b - Enable
17 REQ81_EN1	This register is used to enable and disable LP_FLEXCOMM6 receive request. 0b - Disable 1b - Enable
16 REQ80_EN1	This register is used to enable and disable LP_FLEXCOMM5 transmit request. 0b - Disable 1b - Enable
15 REQ79_EN1	This register is used to enable and disable LP_FLEXCOMM5 receive request. 0b - Disable 1b - Enable
14 REQ78_EN1	This register is used to enable and disable LP_FLEXCOMM4 transmit request. 0b - Disable 1b - Enable
13 REQ77_EN1	This register is used to enable and disable LP_FLEXCOMM4 receive request. 0b - Disable 1b - Enable
12 REQ76_EN1	This register is used to enable and disable LP_FLEXCOMM3 transmit request. 0b - Disable 1b - Enable
11 REQ75_EN1	This register is used to enable and disable LP_FLEXCOMM3 receive request. 0b - Disable 1b - Enable
10 REQ74_EN1	This register is used to enable and disable LP_FLEXCOMM2 transmit request. 0b - Disable 1b - Enable
9 REQ73_EN1	This register is used to enable and disable LP_FLEXCOMM2 receive request. 0b - Disable 1b - Enable
8 REQ72_EN1	This register is used to enable and disable LP_FLEXCOMM1 transmit request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ71_EN1	This register is used to enable and disable LP_FLEXCOMM1 receive request. 0b - Disable 1b - Enable
6 REQ70_EN1	This register is used to enable and disable LP_FLEXCOMM0 transmit request. 0b - Disable 1b - Enable
5 REQ69_EN1	This register is used to enable and disable LP_FLEXCOMM0 receive request. 0b - Disable 1b - Enable
4 REQ68_EN1	This register is used to enable and disable FlexIO0 Shifter7 Status DMA request OR Timer7 Status DMA request. 0b - Disable 1b - Enable
3 REQ67_EN1	This register is used to enable and disable FlexIO0 Shifter6 Status DMA request OR Timer6 Status DMA request. 0b - Disable 1b - Enable
2 REQ66_EN1	This register is used to enable and disable FlexIO0 Shifter5 Status DMA request OR Timer5 Status DMA request. 0b - Disable 1b - Enable
1 REQ65_EN1	This register is used to enable and disable FlexIO0 Shifter4 Status DMA request OR Timer4 Status DMA request. 0b - Disable 1b - Enable
0 REQ64_EN1	This register is used to enable and disable FlexIO0 Shifter3 Status DMA request OR Timer3 Status DMA request. 0b - Disable 1b - Enable

26.5.1.80 DMA1 Request Enable2 (DMA1_REQ_ENABLE2_SET)

Offset

Register	Offset
DMA1_REQ_ENABLE2_SET	7A4h

Function
Writing a 1 to a bit in this register sets the corresponding bit in DMA1_REQ_ENABLE2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...	REQ90_E...	REQ89_E...	REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ95_EN1	Writing a 1 to REQ95_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
30 REQ94_EN1	Writing a 1 to REQ94_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
29 REQ93_EN1	Writing a 1 to REQ93_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
28 REQ92_EN1	Writing a 1 to REQ92_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
27 REQ91_EN1	Writing a 1 to REQ91_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
26	Writing a 1 to REQ90_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ90_EN1	
25 REQ89_EN1	Writing a 1 to REQ89_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
24 REQ88_EN1	Writing a 1 to REQ88_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
23 REQ87_EN1	Writing a 1 to REQ87_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
22 REQ86_EN1	Writing a 1 to REQ86_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
21 REQ85_EN1	Writing a 1 to REQ85_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
20 REQ84_EN1	Writing a 1 to REQ84_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
19 REQ83_EN1	Writing a 1 to REQ83_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
18 REQ82_EN1	Writing a 1 to REQ82_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
17 REQ81_EN1	Writing a 1 to REQ81_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
16 REQ80_EN1	Writing a 1 to REQ80_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
15 REQ79_EN1	Writing a 1 to REQ79_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
14 REQ78_EN1	Writing a 1 to REQ78_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
13 REQ77_EN1	Writing a 1 to REQ77_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
12 REQ76_EN1	Writing a 1 to REQ876_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 REQ75_EN1	Writing a 1 to REQ75_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
10 REQ74_EN1	Writing a 1 to REQ74_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
9 REQ73_EN1	Writing a 1 to REQ73_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
8 REQ72_EN1	Writing a 1 to REQ72_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
7 REQ71_EN1	Writing a 1 to REQ71_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
6 REQ70_EN1	Writing a 1 to REQ70_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
5 REQ69_EN1	Writing a 1 to REQ69_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
4 REQ68_EN1	Writing a 1 to REQ68_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
3 REQ67_EN1	Writing a 1 to REQ67_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
2 REQ66_EN1	Writing a 1 to REQ66_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
1 REQ65_EN1	Writing a 1 to REQ65_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.
0 REQ64_EN1	Writing a 1 to REQ64_EN1 in this register sets the corresponding bit in DMA1_REQ_ENABLE2.

26.5.1.81 DMA1 Request Enable2 (DMA1_REQ_ENABLE2_CLR)

Offset

Register	Offset
DMA1_REQ_ENABLE2_CLR	7A8h

Function

Writing a 1 to a bit in this register clears the corresponding bit in DMA1_REQ_ENABLE2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...	REQ90_E...	REQ89_E...	REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ95_EN1	Writing a 1 to REQ95_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
30 REQ94_EN1	Writing a 1 to REQ94_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
29 REQ93_EN1	Writing a 1 to REQ93_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
28 REQ92_EN1	Writing a 1 to REQ92_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
27 REQ91_EN1	Writing a 1 to REQ91_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
26 REQ90_EN1	Writing a 1 to REQ90_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
25 REQ89_EN1	Writing a 1 to REQ89_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
24 REQ88_EN1	Writing a 1 to REQ88_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
23	Writing a 1 to REQ87_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ87_EN1	
22 REQ86_EN1	Writing a 1 to REQ86_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
21 REQ85_EN1	Writing a 1 to REQ85_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
20 REQ84_EN1	Writing a 1 to REQ84_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
19 REQ83_EN1	Writing a 1 to REQ83_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
18 REQ82_EN1	Writing a 1 to REQ82_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
17 REQ81_EN1	Writing a 1 to REQ81_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
16 REQ80_EN1	Writing a 1 to REQ80_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
15 REQ79_EN1	Writing a 1 to REQ79_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
14 REQ78_EN1	Writing a 1 to REQ78_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
13 REQ77_EN1	Writing a 1 to REQ77_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
12 REQ76_EN1	Writing a 1 to REQ76_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
11 REQ75_EN1	Writing a 1 to REQ75_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
10 REQ74_EN1	Writing a 1 to REQ74_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
9 REQ73_EN1	Writing a 1 to REQ73_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 REQ72_EN1	Writing a 1 to REQ72_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
7 REQ71_EN1	Writing a 1 to REQ71_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
6 REQ70_EN1	Writing a 1 to REQ70_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
5 REQ69_EN1	Writing a 1 to REQ69_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
4 REQ68_EN1	Writing a 1 to REQ68_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
3 REQ67_EN1	Writing a 1 to REQ67_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
2 REQ66_EN1	Writing a 1 to REQ66_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
1 REQ65_EN1	Writing a 1 to REQ65_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.
0 REQ64_EN1	Writing a 1 to REQ64_EN1 in this register clears the corresponding bit in DMA1_REQ_ENABLE2.

26.5.1.82 DMA1 Request Enable2 (DMA1_REQ_ENABLE2_TOG)

Offset

Register	Offset
DMA1_REQ_ENABLE2_TOG	7ACh

Function

Writing a 1 to a bit in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	REQ95_E...	REQ94_E...	REQ93_E...	REQ92_E...	REQ91_E...	REQ90_E...	REQ89_E...	REQ88_E...	REQ87_E...	REQ86_E...	REQ85_E...	REQ84_E...	REQ83_E...	REQ82_E...	REQ81_E...	REQ80_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ79_E...	REQ78_E...	REQ77_E...	REQ76_E...	REQ75_E...	REQ74_E...	REQ73_E...	REQ72_E...	REQ71_E...	REQ70_E...	REQ69_E...	REQ68_E...	REQ67_E...	REQ66_E...	REQ65_E...	REQ64_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQ95_EN1	Writing a 1 to REQ95_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
30 REQ94_EN1	Writing a 1 to REQ94_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
29 REQ93_EN1	Writing a 1 to REQ93_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
28 REQ92_EN1	Writing a 1 to REQ92_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
27 REQ91_EN1	Writing a 1 to REQ91_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
26 REQ90_EN1	Writing a 1 to REQ90_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
25 REQ89_EN1	Writing a 1 to REQ89_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
24 REQ88_EN1	Writing a 1 to REQ88_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
23 REQ87_EN1	Writing a 1 to REQ87_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
22	Writing a 1 to REQ86_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ86_EN1	
21 REQ85_EN1	Writing a 1 to REQ85_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
20 REQ84_EN1	Writing a 1 to REQ84_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
19 REQ83_EN1	Writing a 1 to REQ83_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
18 REQ82_EN1	Writing a 1 to REQ82_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
17 REQ81_EN1	Writing a 1 to REQ81_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
16 REQ80_EN1	Writing a 1 to REQ80_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
15 REQ79_EN1	Writing a 1 to REQ79_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
14 REQ78_EN1	Writing a 1 to REQ78_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
13 REQ77_EN1	Writing a 1 to REQ77_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
12 REQ76_EN1	Writing a 1 to REQ76_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
11 REQ75_EN1	Writing a 1 to REQ75_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
10 REQ74_EN1	Writing a 1 to REQ74_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
9 REQ73_EN1	Writing a 1 to REQ73_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
8 REQ72_EN1	Writing a 1 to REQ72_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 REQ71_EN1	Writing a 1 to REQ71_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
6 REQ70_EN1	Writing a 1 to REQ70_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
5 REQ69_EN1	Writing a 1 to REQ69_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
4 REQ68_EN1	Writing a 1 to REQ68_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
3 REQ67_EN1	Writing a 1 to REQ67_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
2 REQ66_EN1	Writing a 1 to REQ66_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
1 REQ65_EN1	Writing a 1 to REQ65_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.
0 REQ64_EN1	Writing a 1 to REQ64_EN1 in this register toggles the corresponding bit in DMA1_REQ_ENABLE2.

26.5.1.83 DMA1 Request Enable3 (DMA1_REQ_ENABLE3)

Offset

Register	Offset
DMA1_REQ_ENABLE3	7B0h

Function

DMA request 96-121 enable for DMA1. One bit per request. 0: DMA request to DMA1 and response from DMA1 are blocked. 1:DMA request and response are enabled for DMA1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	REQ121_...	REQ120_...	REQ119_...	REQ118_...	REQ117_...	REQ116_...	REQ115_...	REQ114_...	REQ113_...	REQ112_...
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REQ111_...	REQ110_...	REQ109_...	REQ108_...	REQ107_...	REQ106_...	REQ105_...	REQ104_...	REQ103_...	REQ102_...	REQ101_...	REQ100_...	REQ99_E...	REQ98_E...	REQ97_E...	REQ96_E...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 REQ121_EN1	This register is used to enable and disable TSI0 out of range request. 0b - Disable 1b - Enable
24 REQ120_EN1	This register is used to enable and disable TSI0 end of scan request. 0b - Disable 1b - Enable
23 REQ119_EN1	This register is used to enable and disable GPIO5 pin event request 1. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 REQ118_EN1	This register is used to enable and disable GPIO5 pin event request 0. 0b - Disable 1b - Enable
21 REQ117_EN1	This register is used to enable and disable GPIO4 pin event request 1. 0b - Disable 1b - Enable
20 REQ116_EN1	This register is used to enable and disable GPIO4 pin event request 0. 0b - Disable 1b - Enable
19 REQ115_EN1	This register is used to enable and disable GPIO3 pin event request 1. 0b - Disable 1b - Enable
18 REQ114_EN1	This register is used to enable and disable GPIO3 pin event request 0. 0b - Disable 1b - Enable
17 REQ113_EN1	This register is used to enable and disable GPIO2 pin event request 1. 0b - Disable 1b - Enable
16 REQ112_EN1	This register is used to enable and disable GPIO2 pin event request 0. 0b - Disable 1b - Enable
15 REQ111_EN1	This register is used to enable and disable GPIO1 pin event request 1. 0b - Disable 1b - Enable
14 REQ110_EN1	This register is used to enable and disable GPIO1 pin event request 0. 0b - Disable 1b - Enable
13 REQ109_EN1	This register is used to enable and disable GPIO0 pin event request 1. 0b - Disable 1b - Enable
12 REQ108_EN1	This register is used to enable and disable GPIO0 pin event request 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
11 REQ107_EN1	This register is used to enable and disable SINC0 ipd_req_sinc[4] or ipd_req_alt [4] request. 0b - Disable 1b - Enable
10 REQ106_EN1	This register is used to enable and disable SINC0 ipd_req_sinc[3] or ipd_req_alt [3] request. 0b - Disable 1b - Enable
9 REQ105_EN1	This register is used to enable and disable SINC0 ipd_req_sinc[2] or ipd_req_alt [2] request. 0b - Disable 1b - Enable
8 REQ104_EN1	This register is used to enable and disable SINC0 ipd_req_sinc[1] or ipd_req_alt [1] request. 0b - Disable 1b - Enable
7 REQ103_EN1	This register is used to enable and disable SINC0 ipd_req_sinc[0] or ipd_req_alt [0] request. 0b - Disable 1b - Enable
6 REQ102_EN1	This register is used to enable and disable SAI1 transmit request. 0b - Disable 1b - Enable
5 REQ101_EN1	This register is used to enable and disable SAI1 receive request. 0b - Disable 1b - Enable
4 REQ100_EN1	This register is used to enable and disable SAI0 transmit request. 0b - Disable 1b - Enable
3 REQ99_EN1	This register is used to enable and disable SAI0 receive request. 0b - Disable 1b - Enable
2 REQ98_EN1	This register is used to enable and disable I3C1 transmit request. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 REQ97_EN1	This register is used to enable and disable I3C1 receive request. 0b - Disable 1b - Enable
0 REQ96_EN1	This register is used to enable and disable I3C0 transmit request. 0b - Disable 1b - Enable

26.5.1.84 DMA1 Request Enable3 (DMA1_REQ_ENABLE3_SET)

Offset

Register	Offset
DMA1_REQ_ENABLE3_SET	7B4h

Function

Writing a 1 to a bit in this register sets the corresponding bit in DMA1_REQ_ENABLE3

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0										
W							REQ121_...	REQ120_...	REQ119_...	REQ118_...	REQ117_...	REQ116_...	REQ115_...	REQ114_...	REQ113_...	REQ112_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ111_...	REQ110_...	REQ109_...	REQ108_...	REQ107_...	REQ106_...	REQ105_...	REQ104_...	REQ103_...	REQ102_...	REQ101_...	REQ100_...	REQ99_E...	REQ98_E...	REQ97_E...	REQ96_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 REQ121_EN1	Writing a 1 to REQ121_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
24 REQ120_EN1	Writing a 1 to REQ120_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
23 REQ119_EN1	Writing a 1 to REQ119_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
22 REQ118_EN1	Writing a 1 to REQ118_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
21 REQ117_EN1	Writing a 1 to REQ117_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
20 REQ116_EN1	Writing a 1 to REQ116_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
19 REQ115_EN1	Writing a 1 to REQ115_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
18 REQ114_EN1	Writing a 1 to REQ114_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
17 REQ113_EN1	Writing a 1 to REQ113_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
16 REQ112_EN1	Writing a 1 to REQ112_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 REQ111_EN1	Writing a 1 to REQ111_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
14 REQ110_EN1	Writing a 1 to REQ110_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
13 REQ109_EN1	Writing a 1 to REQ109_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
12 REQ108_EN1	Writing a 1 to REQ108_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
11 REQ107_EN1	Writing a 1 to REQ107_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
10 REQ106_EN1	Writing a 1 to REQ106_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
9 REQ105_EN1	Writing a 1 to REQ105_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
8 REQ104_EN1	Writing a 1 to REQ104_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
7 REQ103_EN1	Writing a 1 to REQ103_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
6 REQ102_EN1	Writing a 1 to REQ102_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
5 REQ101_EN1	Writing a 1 to REQ101_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
4 REQ100_EN1	Writing a 1 to REQ100_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
3 REQ99_EN1	Writing a 1 to REQ99_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
2 REQ98_EN1	Writing a 1 to REQ98_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3
1	Writing a 1 to REQ97_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ97_EN1	
0 REQ96_EN1	Writing a 1 to REQ96_EN1 in this register sets the corresponding bit in DMA0_REQ_ENABLE3

26.5.1.85 DMA1 Request Enable3 (DMA1_REQ_ENABLE3_CLR)

Offset

Register	Offset
DMA1_REQ_ENABLE3_CLR	7B8h

Function

Writing a 1 to a bit in this register clears the corresponding bit in DMA1_REQ_ENABLE3

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0										
W							REQ121_...	REQ120_...	REQ119_...	REQ118_...	REQ117_...	REQ116_...	REQ115_...	REQ114_...	REQ113_...	REQ112_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	REQ111_...	REQ110_...	REQ109_...	REQ108_...	REQ107_...	REQ106_...	REQ105_...	REQ104_...	REQ103_...	REQ102_...	REQ101_...	REQ100_...	REQ99_E...	REQ98_E...	REQ97_E...	REQ96_E...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 REQ121_EN1	Writing a 1 to REQ121_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3.
24 REQ120_EN1	Writing a 1 to REQ120_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
23 REQ119_EN1	Writing a 1 to REQ119_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
22 REQ118_EN1	Writing a 1 to REQ118_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
21 REQ117_EN1	Writing a 1 to REQ117_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
20 REQ116_EN1	Writing a 1 to REQ116_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
19 REQ115_EN1	Writing a 1 to REQ115_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
18 REQ114_EN1	Writing a 1 to REQ114_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
17 REQ113_EN1	Writing a 1 to REQ113_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
16 REQ112_EN1	Writing a 1 to REQ112_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
15 REQ111_EN1	Writing a 1 to REQ111_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
14 REQ110_EN1	Writing a 1 to REQ110_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
13 REQ109_EN1	Writing a 1 to REQ109_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
12 REQ108_EN1	Writing a 1 to REQ108_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
11 REQ107_EN1	Writing a 1 to REQ107_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
10 REQ106_EN1	Writing a 1 to REQ106_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
9 REQ105_EN1	Writing a 1 to REQ105_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
8 REQ104_EN1	Writing a 1 to REQ104_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
7 REQ103_EN1	Writing a 1 to REQ103_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
6 REQ102_EN1	Writing a 1 to REQ102_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
5 REQ101_EN1	Writing a 1 to REQ101_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
4 REQ100_EN1	Writing a 1 to REQ100_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
3 REQ99_EN1	Writing a 1 to REQ99_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
2 REQ98_EN1	Writing a 1 to REQ98_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
1 REQ97_EN1	Writing a 1 to REQ97_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3
0	Writing a 1 to REQ96_EN1 in this register clears the corresponding bit in DMA0_REQ_ENABLE3

Table continues on the next page...

Table continued from the previous page...

Field	Function
REQ96_EN1	

Chapter 27
Semaphores2 (SEMA42)

27.1 Chip-specific SEMA42 information

Table 308. Reference links to related information

Topic	Related module	Reference
Full description	SEMA42	SEMA42
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

The AHB ID in AHB bus matrix ports table is the Domain ID.

27.1.1 Module instances

This device contains one instance of the Semaphore module, SEMA42_0.

27.2 Overview

SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single - write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

27.2.1 Block diagram

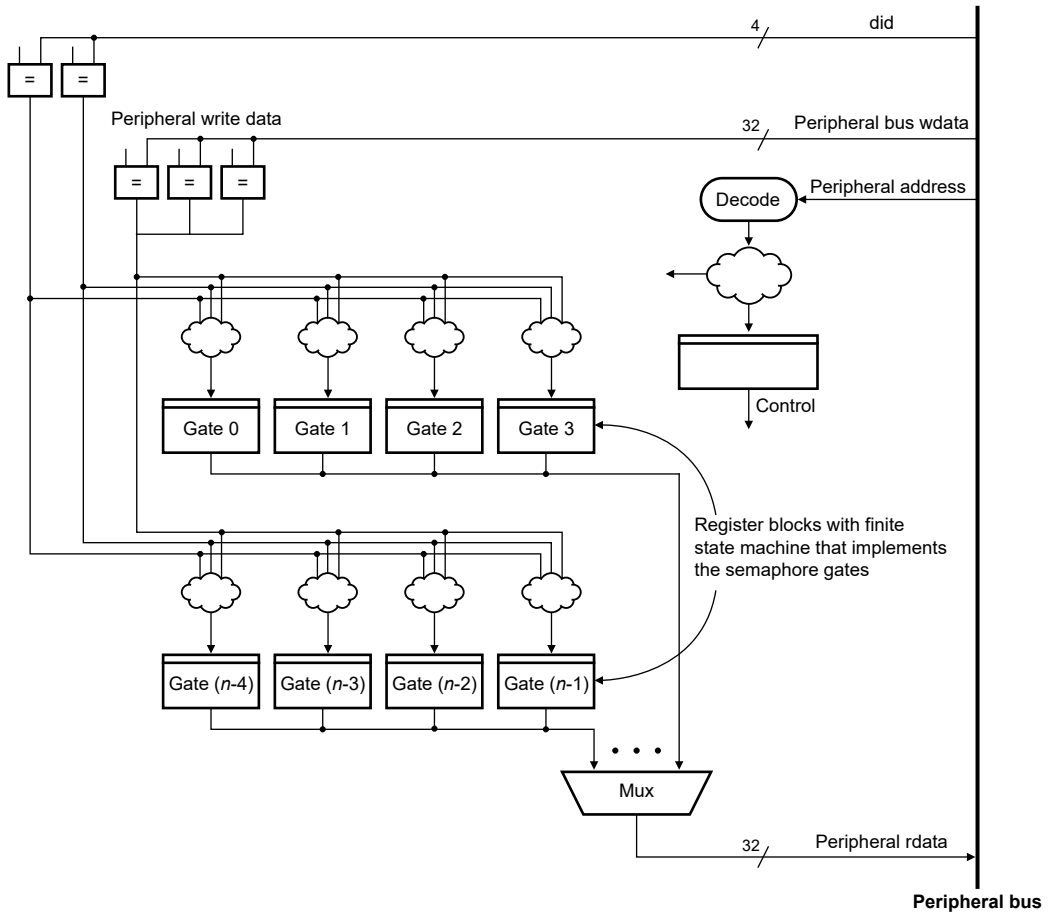


Figure 120. SEMA42 block diagram

27.2.2 Features

SEMA42 implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Support for 16 hardware-enforced gates in a multi-domain configuration that supports up to 15 domains.
- Each hardware gate appears as a 16-state, 4-bit state machine.
- Support for secure reset mechanisms to clear the contents of individual gates, as well as a clear-all capability.
- Memory-mapped slave peripheral that offers programming-model accesses.

27.3 Functional description

The intent of the hardware gate implementation is to specify a protocol where the locking domain must unlock the gate. However, some systems may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset. To support this special gate reset requirement, SEMA42 implements a secure reset mechanism that allows you to initialize a hardware gate (or all the gates) by following a specific dual-write access pattern. The secure-gate reset:

- Uses a technique similar to that required for the servicing of a software watchdog timer
- Requires two consecutive writes with pre-defined data patterns from the same domain

You must do this to force the clearing of the specified gate(s). The required access pattern as follows:

1. A domain performs a 16-bit write to the RSTGT memory location. The most significant byte (RSTGT_W[RSTGDP]) must be E2h. The value of least significant byte is irrelevant for this reference and can be anything.
2. The same domain then performs a second 16-bit write to the RSTGT location. For this write, the upper byte (RSTGT_W[RSTGDP]) is the logical complement of the first data pattern (1Dh) and the lower byte (RSTGT_W[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate or all gates to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, SEMA42 aborts the special gate reset sequence and does not assert an error signal.
3. Reads of the RSTGT location return information on the 2-bit reset state machine (RSTGT_R[RSTGSM]) that implements these functions:
   - The domain performing the reset (RSTGT_R[RSTGMS])
   - The last-cleared gate number(s) (RSTGT_R[RSTGTN])

   Reads of the RSTGT register do not affect the secure-reset finite state machine in any manner.

27.3.1 Multi-core programming: software gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism for system software to control access to shared data structures, shared hardware resources, and so on. The software uses the gating mechanisms to serialize (and synchronize) accesses to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and domains.

Consider the following description of a typical use-case: domain*X* enters a section of code, where shared data values are to be updated. The domain must first acquire a semaphore. Think of this as the locking (or closing) of a software gate. After the gate locks, a properly-architected software system does not allow other processes (or domains) to execute the same code segment or modify the shared data structure protected by the gate. In other words, the system locks out other processes/domains. Many software implementations include a spin-wait loop within the lock function until the gate locks. After domain X obtains the lock, domain X continues execution and updates the data values protected by the particular lock. After domain X completes the updates, it unlocks (or opens) the software gate, allowing other processes/domains access to the updated data values.

A correctly-implemented system solution must follow these important rules:

- A gate variable must protect all writes to shared data values or shared hardware resources.
- After a domain locks a gate, the system must block other processes/domains from accessing the shared data or resources. Software conventions enforce this.
- The domain that locks a particular gate is the only domain that can open (unlock) that gate.

Information in the hardware gate identifying the locking domain can be extremely useful for system-level debugging.

27.3.2 16 Hardware-enforced gates

Gates appear as a 16-entry byte-size array with read and write accesses.

Domains lock gates by writing "domainID_number+1" to the appropriate gate and must read back the gate value to verify that the lock operation succeeded.

After the gate locks, the locking domain unlocks the gate by writing zeroes.

- 16-state implementation

  If gate = 0h, then state = unlocked

  if gate = 1h, then state = locked by domain (master) 0

  if gate = 2h, then state = locked by domain (master) 1

  …

  if gate = Fh, then state = locked by domain (master) 14

SEMA42 uses the logical domain number and the specified data patterns as reference attributes to validate all write operation.

After a gate locks, the locking domain must unlock the gate by writing zeroes.

27.3.3 State machine of the GATE*n* registers

This section describes more about the SEMA42 functional operation and specific details of the state machines of the GATE*n* registers.

As described previously, each of the GATE*n* registers implements a 4-bit, 16-state machine. The following figure shows a simplified diagram of the state transitions for each gate.

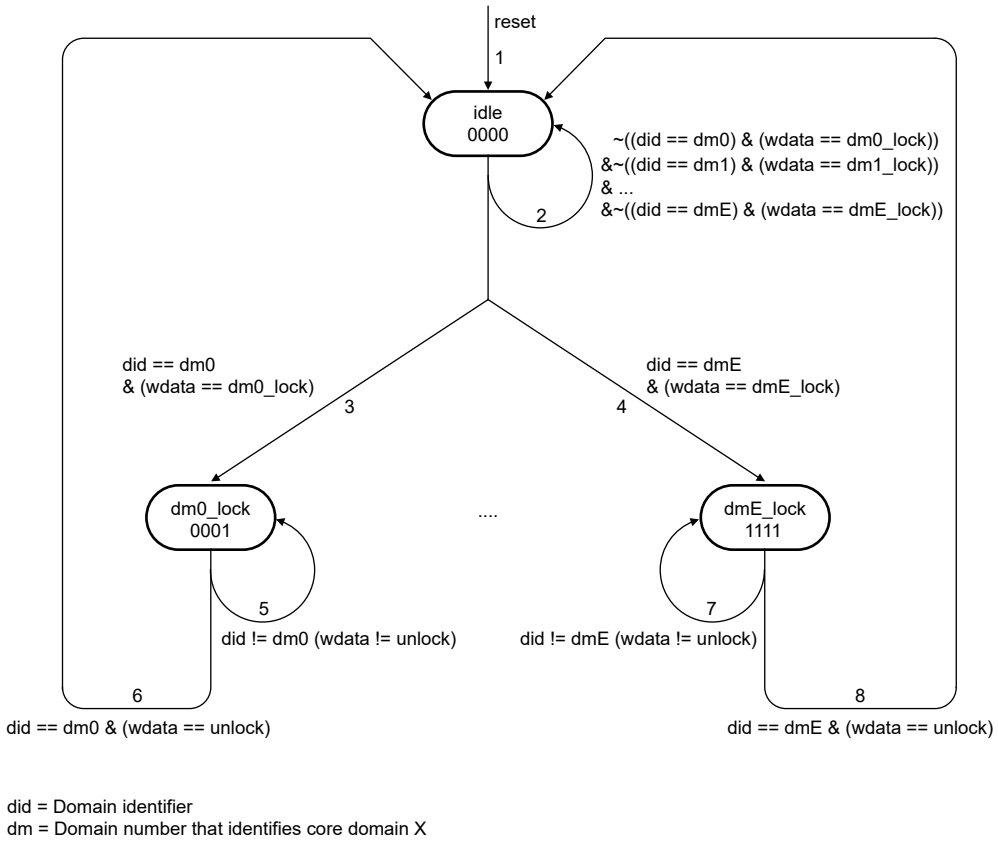


Figure 121. GATE*n* state machine

In the figure above, "dmE" represents domain 14 (E in hexadecimal). The platform passes the domain number to SEMA42.

The following table defines the GATE*n* state transitions.

Table 309. GATE*n* state transitions

Current state	Next state	Transition	Description
–	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding domain occurs, the gate remains in the idle state.
idle	dm0_lock	3	When domain 0h initiates a write of the dm0_lock data value, the gate transitions into the dm0_lock state.

Table continues on the next page...

Table 309. GATE*n* state transitions (continued)

Current state	Next state	Transition	Description
idle	dmE_lock	4	When domain Eh initiates a write of the dmE_lock value, the gate transitions into the dmE_lock state.
dm0_lock	dm0_lock	5	When in this state, the gate remains here if any attempted write is not from domain 0h with the unlock data value.
dm0_lock	idle	6	The gate returns to the idle (unlocked) state after a write from domain 0h with the unlock data value occurs.
dmE_lock	dmE_lock	7	When in this state, the gate remains here if any attempted write is not from domain Eh with the unlock data value.
dmE_lock	idle	8	The gate returns to the idle (unlocked) state after a write from domain Eh with the unlock data value occurs.

SEMA42 uses these gate data values:

- The lock data value is (domain number) + 1.
- The unlock data value is 00h.

27.3.4 Clocking

Type of clock	Description
Module clock (clk)	Functions the module and controls the access to the registers.

27.3.5 Interrupts

This module has no interrupts.

27.4 Memory map/register definition

You can access these registers only in Supervisor mode. User accesses terminate with an error.

27.4.1 SEMA42 register descriptions

27.4.1.1 SEMA42 memory map

SEMA42_0 base address: 400B_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h - Fh	Gate (GATE0 - GATE15) [1]	8	RW	00h
42h	Reset Gate Read (RSTGT_R)	16	R	0000h
42h	Reset Gate Write (RSTGT_W)	16	W	See section

1. In this array, the index and offset values of the registers do not increment in direct alignment. For details, see the register description.

27.4.1.2 Gate (GATE0 - GATE15)

Offset

For n = 0 to 15:

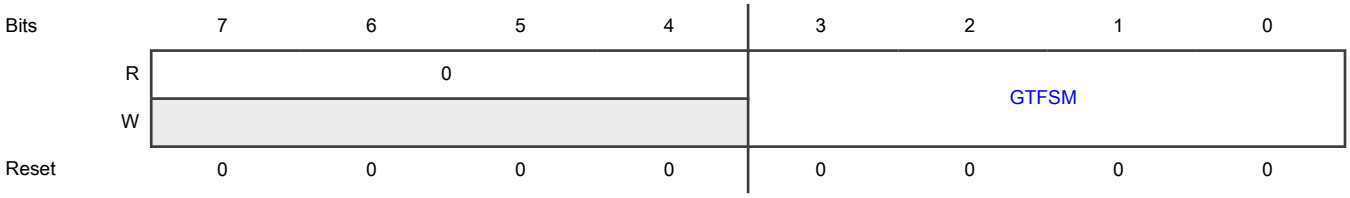
Register	Offset
GATEn	0h + (n + 3 - 2 × (n mod 4))

Function

Implements each semaphore gate in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical domain-identifier number in conjunction with the data patterns to validate all attempted write operations. Only domain masters can modify the gate registers. After a gate locks, only the locking domain must open (unlock) the gate.

You can read multiple gate values in a single access. However, you can update only a single gate at a time, via a write operation. If you attempt to write a byte-wide value that is neither the unlock value (00h) nor the appropriate lock value (domainID_number + 1), SEMA42 considers this as "no operation" and does not change any gate state. Attempts to write multiple gates in a single-aligned access with a size larger than 8 bits (byte) generate an error termination and do not allow any gate state changes.

Diagram



Fields

Field	Function
7-4 —	Reserved
3-0 GTFSM	Gate Finite State Machine Indicates the state of the gate for the last domain that locked the gate. This can be useful during system debug. The hardware gate has a 16-state implementation, defined as: 0000b - The gate is unlocked (free). 0001b - Domain 0 locked the gate. 0010b - Domain 1 locked the gate. 0011b - Domain 2 locked the gate. 0100b - Domain 3 locked the gate.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0101b - Domain 4 locked the gate. 0110b - Domain 5 locked the gate. 0111b - Domain 6 locked the gate. 1000b - Domain 7 locked the gate. 1001b - Domain 8 locked the gate. 1010b - Domain 9 locked the gate. 1011b - Domain 10 locked the gate. 1100b - Domain 11 locked the gate. 1101b - Domain 12 locked the gate. 1110b - Domain 13 locked the gate. 1111b - Domain 14 locked the gate.

27.4.1.3 Reset Gate Read (RSTGT_R)

Offset

Register	Offset
RSTGT_R	42h

Function

Describes the specific hardware gate to be reset and records the logic number of domain. Reset Gate Write (RSTGT_W) also describe the same register showing the fields when you write it.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		RSTGSM		RSTGMS				RSTGTN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 —	Reserved
13-12	Reset Gate Finite State Machine

Table continues on the next page...

Table continued from the previous page...

Field	Function
RSTGSM	Indicates the encoded state machine value when you read the register. RSTGSM = 10b is valid for only a single machine cycle, so a read can never return this value. SEMA42 maintains the reset state machine in a 2-bit, 3-state implementation, defined as follows: 00b - Idle, waiting for the first data pattern write. 01b - Waiting for the second data pattern write 10b - The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. 11b - This state encoding is never used and therefore reserved.
11-8 RSTGMS	Reset Gate Domain Records the logical number of the domain performing the gate reset function. To succeed, this function requires that the same domain initiate the two consecutive writes to this register. SEMA42 updates the field each time a write to this register occurs.
7-0 RSTGTN	Reset Gate Number Specifies the specific hardware gate to be reset. The second write updates this field. If RSTGTN < 64, SEMA42 resets the single gate defined by RSTGTN. Otherwise, SEMA42 resets all the gates.

27.4.1.4 Reset Gate Write (RSTGT_W)

Offset

Register	Offset
RSTGT_W	42h

Function

Specifies the hardware gate to reset when data patterns are specified.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	RSTGDP								RSTGTN							
Reset [1]	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

1. Reset value is not applicable for writes.

Fields

Field	Function
15-8 RSTGDP	Reset Gate Data Pattern You access this field with the specified data patterns on the two consecutive writes to enable the gate-reset mechanism. For the first write, RSTGDP must be E2h. For the second write, RSTGDP must be 1Dh.
7-0 RSTGTN	Reset Gate Number Specifies the specific hardware gate to be reset. The second write updates this field. If RSTGTN < 64, SEMA42 resets the single gate defined by RSTGTN. Otherwise, SEMA42 resets all the gates.

27.5 External signals

This module has no external signals.

27.6 Initialization

This module does not require initialization.

Chapter 28
Interrupt Monitor (INTM)

28.1 Chip-specific INTM information

Table 310. Reference links to related information

Topic	Related module	Reference
Full description	INTM	INTM
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

INTM only monitors the interrupts that are connected to CPU0.

28.1.1 Module instances

This device contains one instance of the Interrupt Monitor module, INTM0.

28.2 Overview

INTM provides a mechanism to monitor the latency of the responses on interrupt requests to ensure that the processing of these critical interrupts executes within the expected time frame, increasing the reliability of the device.

28.2.1 Block diagram

The following block diagram shows the major registers involved in monitoring the interrupt sources.

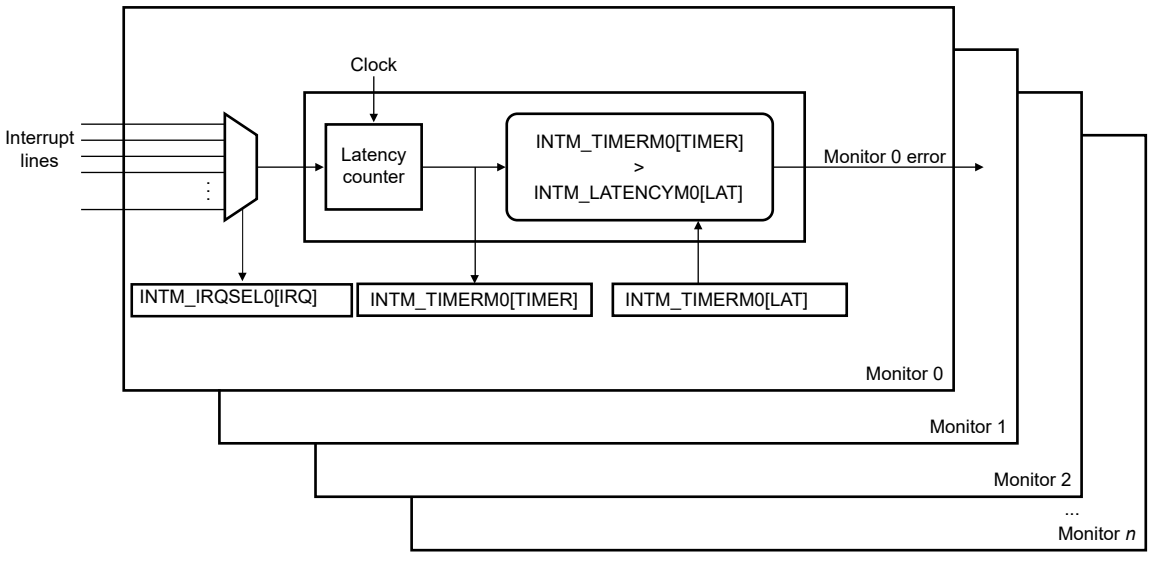


Figure 122. INTM block diagram

28.2.2 Features

INTM supports the following features:

- 4 programmable interrupt monitors
- Programmable interrupt source per monitor
- Programmable 24-bit maximum latency counter per monitor
- Timer expired status bit per monitor
- One interrupt acknowledge for all monitors

28.3 Functional description

Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3) provides selection of the source, Monitor Mode (INTM_MM) enables the monitor, Interrupt Acknowledge (INTM_IACK) captures the event when the interrupt is acknowledged, Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) can be programmed to trigger a monitor error when a threshold value is exceeded, Status for Monitor a (INTM_STATUS0 - INTM_STATUS3) is available to read the monitor error over peripheral bus.

NOTE

INTM supports only level-type interrupts. Hence, the pulse-type interrupts are converted into level-type interrupts for INTM.

28.3.1 Clocking

This module has no clocking considerations.

28.3.2 Interrupts

This module has no interrupts.

28.4 External signals

This module has no external signals.

28.5 Initialization

To monitor a subset of interrupt sources:

1. Program Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3) with a value corresponding to the interrupt source number to be monitored.
   - You can monitor only defined interrupt sources.
2. Program Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) to the maximum expected latency time for each monitored interrupt source.
3. Write 1 to INTM_MM[MM] to monitor the registers.
4. During the interrupt service routine, write the IRQ of the ISR to Interrupt Acknowledge (INTM_IACK).
5. If the maximum expected latency time exceeds the limit defined in Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3), then an error indication is sent to Fault Collection Control Unit(FCCU). You can also read Status for Monitor a (INTM_STATUS0 - INTM_STATUS3) to see this information.

To clear an error condition, write 0 to the corresponding INTM_TIMERa[TIMER] field for which the interrupt source exceeded the programmed latency value. In case the source is unknown, NXP recommends you to write 0 to all INTM_TIMERa[TIMER] fields.

28.6 INTM register descriptions

INTM allows you to set a maximum timer count for the interrupt latency from interrupt request to interrupt acknowledge. This mechanism monitors the latency of interrupt sources to ensure that these critical interrupts execute within the expected time

frame, increasing the reliability of the chip. The hardware for this function is isolated in its own hierarchy to decrease the risk of fault interference.

An error indication is sent to FCCU if the interrupt processing latency exceeds the programmed threshold of the expected latency. The error indication can be read from INTM_STATUSa[STATUS].

28.6.1 INTM memory map

INTM0 base address: 4005_D000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Monitor Mode (INTM_MM)	32	RW	0000_0000h
4h	Interrupt Acknowledge (INTM_IACK)	32	W	0000_0000h
8h	Interrupt Request Select for Monitor 0 (INTM_IRQSEL0)	32	RW	0000_0000h
Ch	Interrupt Latency for Monitor 0 (INTM_LATENCY0)	32	RW	0000_0000h
10h	Timer for Monitor 0 (INTM_TIMER0)	32	RW	0000_0000h
14h	Status for Monitor 0 (INTM_STATUS0)	32	R	0000_0000h
18h	Interrupt Request Select for Monitor 1 (INTM_IRQSEL1)	32	RW	0000_0000h
1Ch	Interrupt Latency for Monitor 1 (INTM_LATENCY1)	32	RW	0000_0000h
20h	Timer for Monitor 1 (INTM_TIMER1)	32	RW	0000_0000h
24h	Status for Monitor 1 (INTM_STATUS1)	32	R	0000_0000h
28h	Interrupt Request Select for Monitor 2 (INTM_IRQSEL2)	32	RW	0000_0000h
2Ch	Interrupt Latency for Monitor 2 (INTM_LATENCY2)	32	RW	0000_0000h
30h	Timer for Monitor 2 (INTM_TIMER2)	32	RW	0000_0000h
34h	Status for Monitor 2 (INTM_STATUS2)	32	R	0000_0000h
38h	Interrupt Request Select for Monitor 3 (INTM_IRQSEL3)	32	RW	0000_0000h
3Ch	Interrupt Latency for Monitor 3 (INTM_LATENCY3)	32	RW	0000_0000h
40h	Timer for Monitor 3 (INTM_TIMER3)	32	RW	0000_0000h
44h	Status for Monitor 3 (INTM_STATUS3)	32	R	0000_0000h

28.6.2 Monitor Mode (INTM_MM)

Offset

Register	Offset
INTM_MM	0h

Function

Enables the cycle count timer on a monitored interrupt request for comparison to the latency register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															MM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 MM	Monitor Mode Controls whether the INTM monitors the latency of an interrupt response. 0b - Disable 1b - Enable

28.6.3 Interrupt Acknowledge (INTM_IACK)

Offset

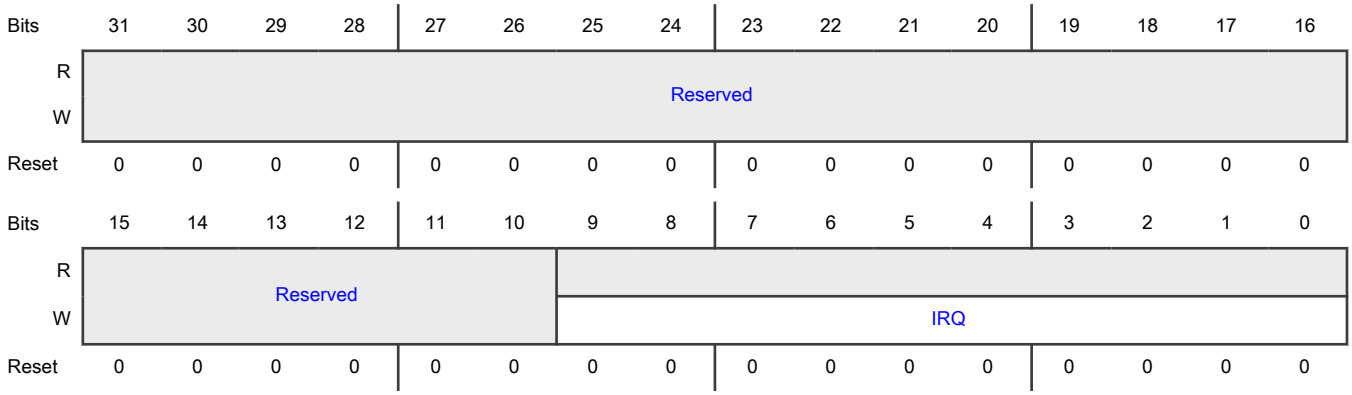
Register	Offset
INTM_IACK	4h

Function

Acknowledges the interrupt processing.

The Interrupt service routine writes to this register after the register acknowledges interrupt processing. This write operation must provide the number of the processed interrupt, which is compared with the interrupt request number in each Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3) and stops the corresponding Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3) when it found a match.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 IRQ	Interrupt Request Specifies the interrupt request number to stop Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3).

28.6.4 Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3)

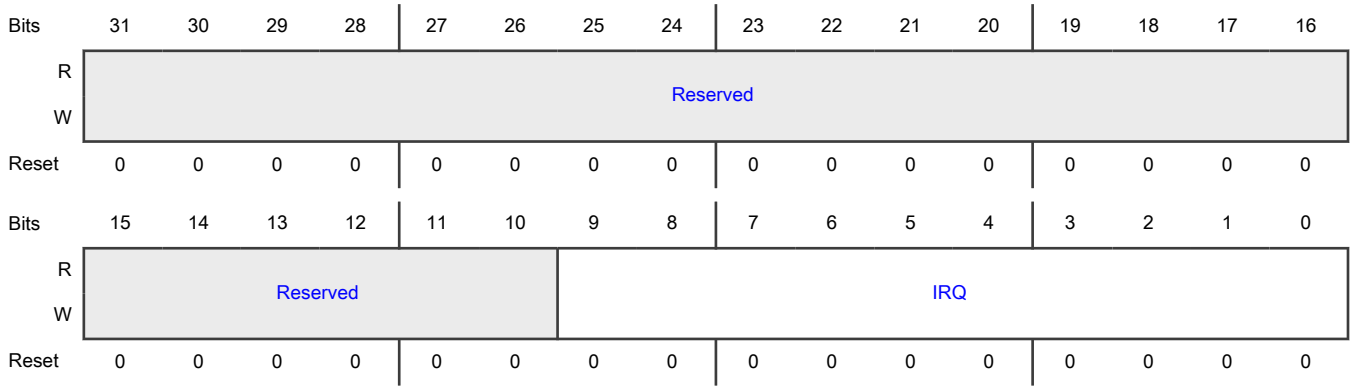
Offset

Register	Offset
INTM_IRQSEL0	8h
INTM_IRQSEL1	18h
INTM_IRQSEL2	28h
INTM_IRQSEL3	38h

Function

Indicates which interrupt request must be monitored or checked.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 IRQ	Interrupt Request Selects the interrupt request number to monitor.

28.6.5 Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3)

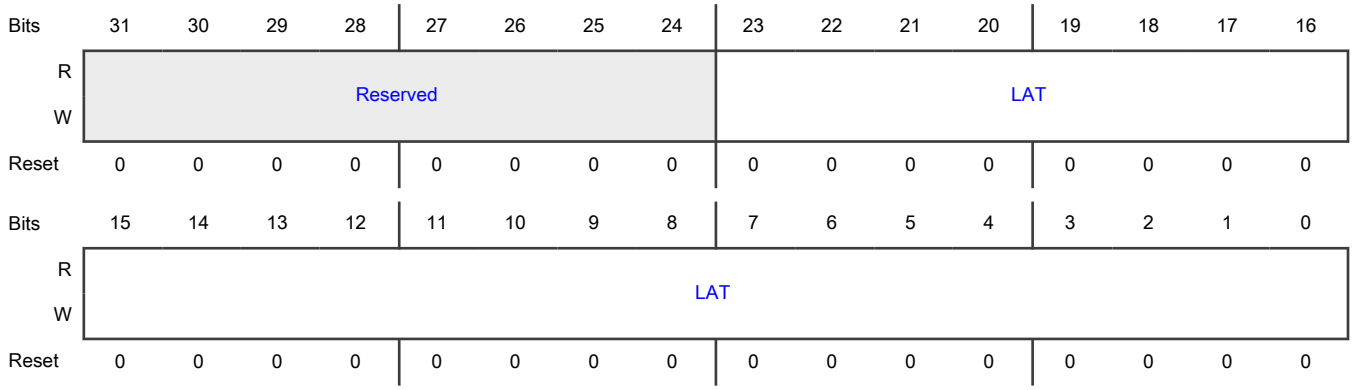
Offset

Register	Offset
INTM_LATENCY0	Ch
INTM_LATENCY1	1Ch
INTM_LATENCY2	2Ch
INTM_LATENCY3	3Ch

Function

Indicates the maximum time before an error signal is asserted for a detected interrupt request to interrupt acknowledge.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 LAT	Latency Specifies the maximum number of INTM clock cycles allowed for the monitored interrupt request. Maximum programmed value must not exceed FF_FFFDh to allow for proper timer error capture.

28.6.6 Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3)

Offset

Register	Offset
INTM_TIMER0	10h
INTM_TIMER1	20h
INTM_TIMER2	30h
INTM_TIMER3	40h

Function

Counts the number of INTM clock cycles for a detected interrupt request to an acknowledged interrupt processing.

Initializes to 1 in following conditions:

- Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) is nonzero
- Monitor error is not asserted
- Zero to one transition on the monitored interrupt request

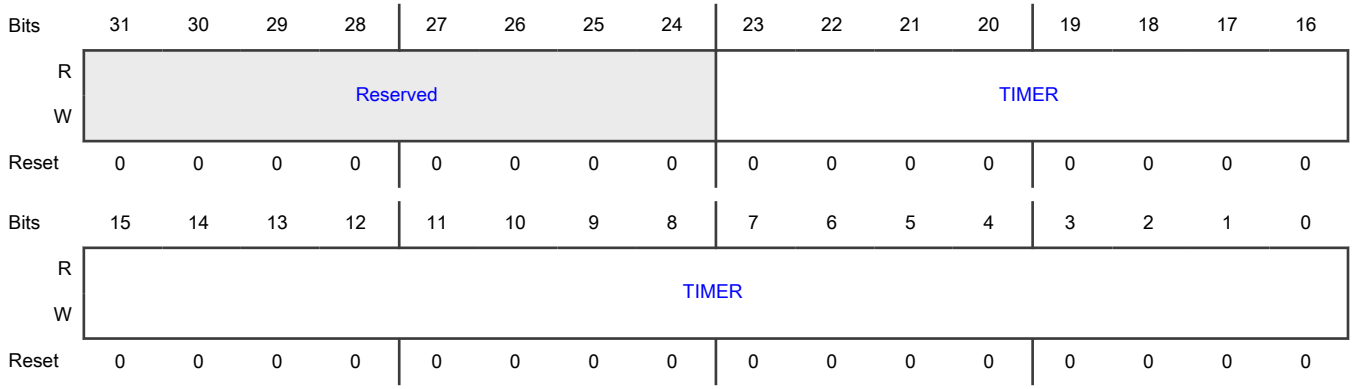
Following are the conditions to enable Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3):

- INTM_MM = 1 and zero-to-one transition on the monitored interrupt source

Following are the conditions to disable Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3):

- Reset
- INTM_MM = 1 and monitor error
- INTM_MM = 1 and monitored interrupt request is not asserted
- Interrupt acknowledge for corresponding interrupt request
- After the timer is enabled, writing 0 to INTM_MM[MM] has no effect.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 TIMER	Timer Counts the number of INTM clock cycles up to 24 bits of resolution.

28.6.7 Status for Monitor a (INTM_STATUS0 - INTM_STATUS3)

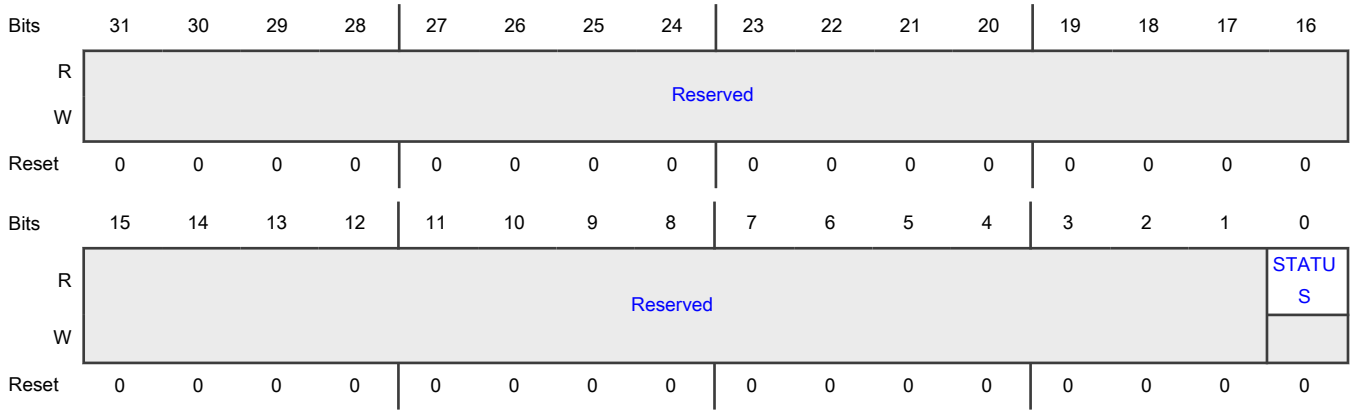
Offset

Register	Offset
INTM_STATUS0	14h
INTM_STATUS1	24h
INTM_STATUS2	34h
INTM_STATUS3	44h

Function

Defines the monitor status. Indicates whether Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3) value has exceeded Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) value. You can clear the monitor status by writing 0 to the corresponding INTM_TIMERa[TIMER].

Diagram



Fields

Field	Function
31-1 —	Reserved
0 STATUS	Monitor status Indicates whether Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3) value has exceeded the Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) value. 0b - Did not exceed 1b - Exceeded

Chapter 29
Error Injection Module (EIM)

29.1 Chip-specific EIM information

Table 311. Reference links to related information

Topic	Related module	Reference
Full description	EIM	EIM
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

EIM is integrated between the memory controller and memory array to enable a controlled way for error injection. Each memory controller has its own EIM channel.

29.1.1 Module instances

This device contains one instance of the Error Injection module, EIM0.

29.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

29.1.3 EIM channel mapping

Table 312. EIM channel mapping

EIM channel #	Target	Data bits of RAM	Check bits of RAM	Word0 number of check bits	Word1 number of data bits
0	RAMX 0	[31:0] RAMX 0 Read Data	[6:0] RAMH 1 read data	7	32
	RAMX 1	[31:0] RAMX 1 Read Data	[14:8] RAMH 1 read data	7	32
	RAMX 2	[31:0] RAMX 2 Read Data	[22:16] RAMH 1 read data	7	32
1	RAMA 0	[31:0] RAMA 0 Read Data	[38:32] RAMA 0 read data ECCbits	7	32
	RAMA 1	[31:0] RAMA 1 Read Data	[38:32] RAMA 1 read data ECC bits	7	32
	RAMA 2	[31:0] RAMA 2 Read Data	[38:32] RAMA 2 read data ECC bits	7	32
	RAMA 3	[31:0] RAMA 3 Read Data	[38:32] RAMA 3 read data ECC bits	7	32

Table continues on the next page...

Table 312. EIM channel mapping (continued)

EIM channel #	Target	Data bits of RAM	Check bits of RAM	Word0 number of check bits	Word1 number of data bits
2	RAMB 0	[31:0] RAMB 0 Read Data	[6:0] RAMH 0 read data	7	32
3	RAMC 0	[31:0] RAMC 0 Read Data	[6:0] RAMG 3 read data	7	32
	RAMC 1	[31:0] RAMC 1 Read Data	[14:8] RAMG 3 read data	7	32
4	RAMD 0	[31:0] RAMD 0 Read Data	[6:0] RAMG 2 read data	7	32
	RAMD 1	[31:0] RAMD 1 Read Data	[14:8] RAMG 2 read data	7	32
5	RAME 0	[31:0] RAME 0 Read Data	[6:0] RAMG 1 read data	7	32
	RAME 1	[31:0] RAME 1 Read Data	[14:8] RAMG 1 read data	7	32
6	RAMF 0	[31:0] RAMF 0 Read Data	[6:0] RAMG 0 read data	7	32
	RAMF 1	[31:0] RAMF 1 Read Data	[14:8] RAMG 0 read data	7	32
7	LPCACRAM	[31:0] LPCAC RAM Read Data	[32] LPCAC RAM readdata Parity bit	1	32
8	PKCRAM	[7:0] PKC RAM 0/1 [16:9] PKC RAM 0/1 [25:18] PKC RAM 0/1 [34:27] PKC RAM 0/1	[8] PKC RAM 0/1 [17] PKC RAM 0/1 [26] PKC RAM 0/1 [35] PKC RAM 0/1	4	32

29.1.4 EIM enable

Two enables are supported; GEIEN which enables error injection globally, EICHnEN which enables error injection functionality for a particular channel. This double layer enable provides protection against accidental enabling and reconfiguration of the error injection function of each channel. The EICHDn_WORD0 register is written to inject error in desired check bits, and the EICHDn_WORD1 register is written to inject error in desired data bits.

29.2 Overview

The Error Injection Module (EIM) is mainly used for diagnostic purposes. It provides a method to test the diagnostics (memory ECC, interconnect parity) by error injection in the field. See the chip-specific EIM information to determine which functional safety features are supported by this method.

EIM enables you to inject artificial errors on error-checking mechanisms of a system, such as ECC for RAM read data and parity bits. For each such mechanism that EIM supports on the chip, EIM can inject single-bit and multi-bit inversions on data in the applicable target bus. Injecting faults on memory accesses can be used to exercise the SEC-DED ECC function of the related system.

29.2.1 Features

The EIM includes these features:

- Supports 9 error injection channels. See the chip-specific EIM information for channel assignment details.
- Protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

29.2.2 Block diagram

The following diagram shows an example of EIM implementation with a 64-bit read data bus and an 8-bit checkbit bus.

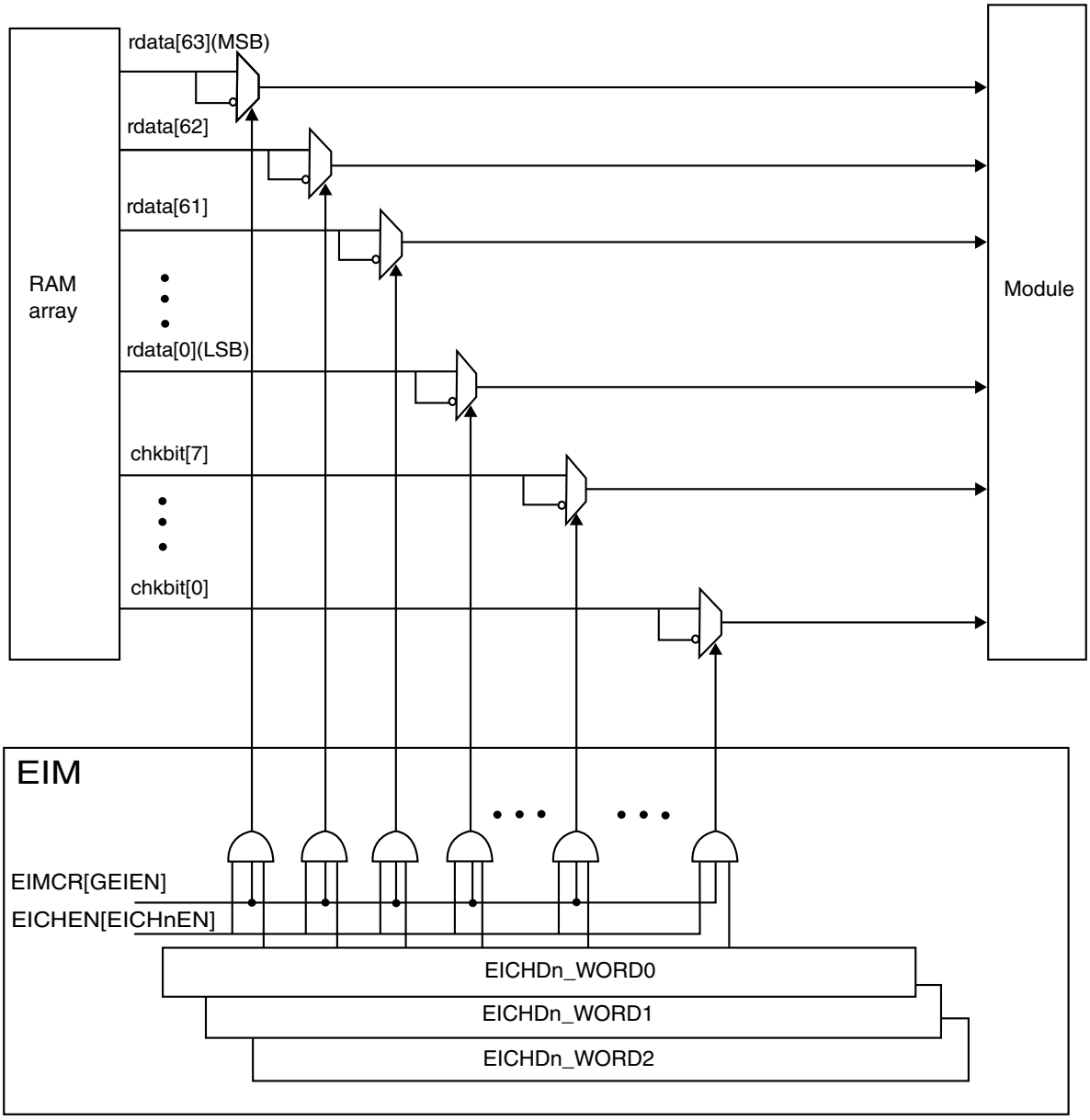


Figure 123. EIM functional block diagram (64-bit read data bus and 8-bit check bit bus)

Several memory elements are implemented within a device, which may not only be the large memory blocks (Flash and SRAM) but also smaller memories like caches, the TCD blocks, and the embedded peripheral memories. Some larger memories may actually be built from multiple memory elements, dependent on their size or function. Each of these memory elements implements its own control logic, the memory controller, that performs the accesses to the actual memory, the memory array. An EIM channel is associated with a memory controller and provides the capability to alter one or multiple signals in the read access path from the corresponding memory array(s). Only memory controllers controlling a safety related memory may be associated with an EIM channel.

29.3 Functional description

The EIM provides protection against accidental enabling and reconfiguration of the error injection function by enforcing a two-stage enablement mechanism. To properly enable the error injection mechanism for a channel:

- Write 1 to the EICHEN[EICH*n*EN] field, where *n* denotes the channel number.
- Write 1 to EIMCR[GEIEN].

NOTE

When the use case for a channel requires writing any EICHD*n*_WORD register, write the EICHD*n*_WORD register before executing the two-stage enablement mechanism. A successful write to any EICHD*n*_WORD register clears the corresponding EICHEN[EICH*n*EN] field.

The EIM supports 9 error injection channels. See the chip-specific EIM information for channel assignment details. Each channel:

- Can be assigned to a single memory array interface by intercepting the assigned memory read data bus and checkbit bus, and injects errors by inverting the value transmitted for selected bits on each bus line.
- Can be assigned to a redundant comparison unit by intercepting the signals being compared, and injecting errors by inverting the value transmitted for selected bits on each bus line.

On a memory read access, the applicable EICHD*n*_WORD registers define which bits of the read data and/or checkbit bus to invert.

Figure 123 depicts the interception and override of a 64-bit read data bus and an 8-bit checkbit data bus for an example memory array.

Error injection scenarios

The EIM supports these cases of error injection:

- To generate a single-bit error, invert only 1 bit of the CHKBIT_MASK or DATA_MASK in the EICHD*n*_WORD registers.
- To generate a multi-bit error, invert only 2 bits of the CHKBIT_MASK or DATA_MASK in the EICHD*n*_WORD registers.

NOTE

An attempt to invert more than 2 bits in one operation might result in undefined behavior.

To enable error injection:

1. Set the EICHD*n*_WORD*m*[CHKBIT_MASK] and EICHD*n*_WORD*m*[B*a*_*b*DATA_MASK] fields for each channel that will be driving an injection.
2. Program the EICHEN register to enable the channels that will be injecting errors.
3. Set the EIMCR[GEIEN] field to globally allow all enabled channels to actively inject errors.

To disable error injection, either disable the EIMCR[GEIEN] field or disable the individual channel enable fields of the EICHEN register.

29.4 Initialization

This module does not require initialization.

29.6 EIM register descriptions

The EIM provides a programming model mapped to an on-platform peripheral slot.

Programming model access

All system bus masters can access the programming model:

- Only in supervisor mode

- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an error termination:

- In user mode
- Using non-32-bit access sizes
- To undefined (reserved) addresses

Attempted updates to the programming model while the EIM is in the midst of an operation result in non-deterministic behavior.

Error injection channel descriptor: function and structure

Each error injection channel descriptor:

- Specifies a mask that defines which bits of the read data and/or checkbit bus from target RAM are inverted on a read access.
- Consists of a 128-bit (16-byte) structure, composed of four 32-bit words, in the EIM programming model. Unused words are not documented.
  - Word0 (EICHD*n*_WORD0), if present, defines the checkbit mask.
  - Word1-3 (EICHD*n*_WORD1-3), if present, define the data mask. Word2 and Word3 are present only when required by the total width of the channel's data mask. See Error injection channel descriptor: DATA_MASK details.

The multiple channel descriptors are organized sequentially.

Error injection channel descriptor: DATA_MASK details

For each channel: The following table shows the total width of DATA_MASK and the distribution of its bits across the WORD registers.

Table 313. Error injection channel descriptor: DATA_MASK details

Channel	DATA_MASK total width (bits)	Specific bits of DATA_MASK in		
		WORD1	WORD2	WORD3
0	32	31-0	—	—
1	32	31-0	—	—
2	32	31-0	—	—
3	32	31-0	—	—
4	32	31-0	—	—
5	32	31-0	—	—
6	32	31-0	—	—
7	32	31-0	—	—
8	32	31-0	—	—

29.6.1 EIM memory map

EIM0 base address: 4005_B000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Error Injection Module Configuration Register (EIMCR)	32	RW	0000_0000h
4h	Error Injection Channel Enable register (EICHEN)	32	RW	0000_0000h
100h	Error Injection Channel Descriptor 0, Word0 (EICHD0_WORD0)	32	RW	0000_0000h
104h	Error Injection Channel Descriptor 0, Word1 (EICHD0_WORD1)	32	RW	0000_0000h
140h	Error Injection Channel Descriptor 1, Word0 (EICHD1_WORD0)	32	RW	0000_0000h
144h	Error Injection Channel Descriptor 1, Word1 (EICHD1_WORD1)	32	RW	0000_0000h
180h	Error Injection Channel Descriptor 2, Word0 (EICHD2_WORD0)	32	RW	0000_0000h
184h	Error Injection Channel Descriptor 2, Word1 (EICHD2_WORD1)	32	RW	0000_0000h
1C0h	Error Injection Channel Descriptor 3, Word0 (EICHD3_WORD0)	32	RW	0000_0000h
1C4h	Error Injection Channel Descriptor 3, Word1 (EICHD3_WORD1)	32	RW	0000_0000h
200h	Error Injection Channel Descriptor 4, Word0 (EICHD4_WORD0)	32	RW	0000_0000h
204h	Error Injection Channel Descriptor 4, Word1 (EICHD4_WORD1)	32	RW	0000_0000h
240h	Error Injection Channel Descriptor 5, Word0 (EICHD5_WORD0)	32	RW	0000_0000h
244h	Error Injection Channel Descriptor 5, Word1 (EICHD5_WORD1)	32	RW	0000_0000h
280h	Error Injection Channel Descriptor 6, Word0 (EICHD6_WORD0)	32	RW	0000_0000h
284h	Error Injection Channel Descriptor 6, Word1 (EICHD6_WORD1)	32	RW	0000_0000h
2C0h	Error Injection Channel Descriptor 7, Word0 (EICHD7_WORD0)	32	RW	0000_0000h
2C4h	Error Injection Channel Descriptor 7, Word1 (EICHD7_WORD1)	32	RW	0000_0000h
300h	Error Injection Channel Descriptor 8, Word0 (EICHD8_WORD0)	32	RW	0000_0000h
304h	Error Injection Channel Descriptor 8, Word1 (EICHD8_WORD1)	32	RW	0000_0000h

29.6.2 Error Injection Module Configuration Register (EIMCR)

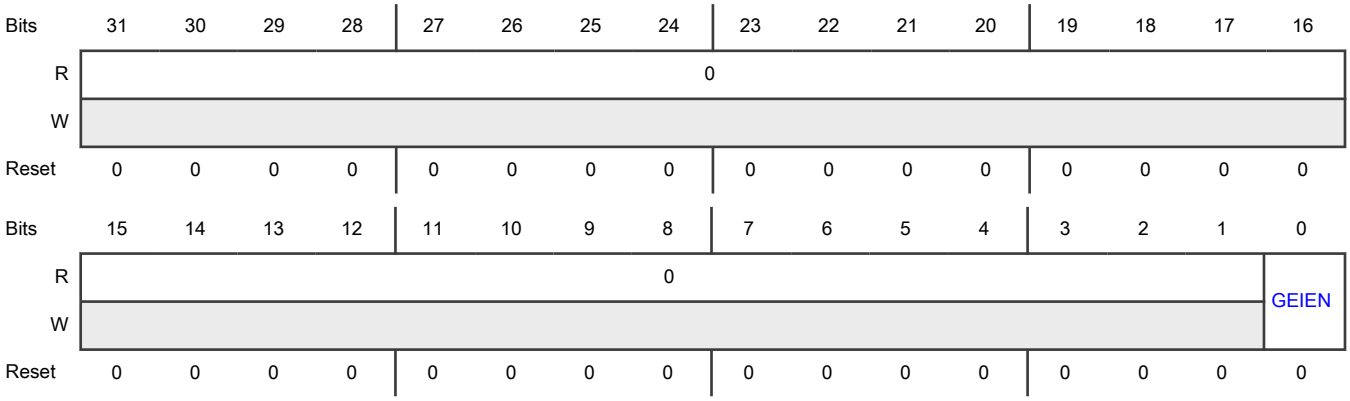
Offset

Register	Offset
EIMCR	0h

Function

The EIM Configuration Register is used to globally enable/disable the error injection function.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 GEIEN	Global Error Injection Enable This bit globally enables or disables the error injection function of the EIM. This field is initialized by hardware reset. 0b - Disabled 1b - Enabled

29.6.3 Error Injection Channel Enable register (EICHEN)

Offset

Register	Offset
EICHEN	4h

Function

Each field of the Error Injection Channel Enable register (EICHEN) is used to enable or disable the corresponding error injection channel.

NOTE

To enable an error injection channel, the Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EICH0EN	EICH1EN	EICH2EN	EICH3EN	EICH4EN	EICH5EN	EICH6EN	EICH7EN	EICH8EN	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 EICH0EN	Error Injection Channel 0 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 0 1b - Error injection is enabled on Error Injection Channel 0
30 EICH1EN	Error Injection Channel 1 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 1 1b - Error injection is enabled on Error Injection Channel 1
29 EICH2EN	Error Injection Channel 2 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 2 1b - Error injection is enabled on Error Injection Channel 2
28 EICH3EN	Error Injection Channel 3 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 3 1b - Error injection is enabled on Error Injection Channel 3
27 EICH4EN	Error Injection Channel 4 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 4 1b - Error injection is enabled on Error Injection Channel 4
26 EICH5EN	Error Injection Channel 5 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 5 1b - Error injection is enabled on Error Injection Channel 5
25 EICH6EN	Error Injection Channel 6 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 6 1b - Error injection is enabled on Error Injection Channel 6
24 EICH7EN	Error Injection Channel 7 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 7 1b - Error injection is enabled on Error Injection Channel 7
23 EICH8EN	Error Injection Channel 8 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. 0b - Error injection is disabled on Error Injection Channel 8 1b - Error injection is enabled on Error Injection Channel 8
22 —	Reserved
21 —	Reserved
20 —	Reserved
19 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 —	Reserved
17 —	Reserved
16 —	Reserved
15 —	Reserved
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved
4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

29.6.4 Error Injection Channel Descriptor n, Word0 (EICHD0_WORD0 - EICHD6_WORD0)

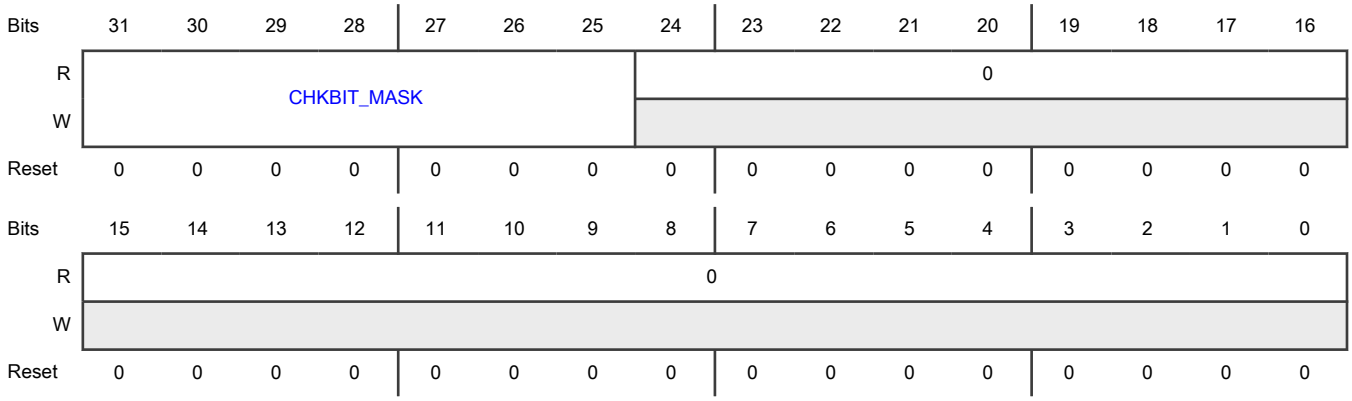
Offset

Register	Offset
EICHD0_WORD0	100h
EICHD1_WORD0	140h
EICHD2_WORD0	180h
EICHD3_WORD0	1C0h
EICHD4_WORD0	200h
EICHD5_WORD0	240h
EICHD6_WORD0	280h

Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

Diagram



Fields

Field	Function
31-25 CHKBIT_MASK	Checkbit Mask This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored. For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information. **NOTE** Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[6:0] (7 bits wide), CHKBIT_MASK[6] is in the position of the most significant bit. 0b - The corresponding bit of the checkbit bus remains unmodified. 1b - The corresponding bit of the checkbit bus is inverted.
24-0 —	Reserved

29.6.5 Error Injection Channel Descriptor n, Word1 (EICHD0_WORD1 - EICHD8_WORD1)

Offset

Register	Offset
EICHD0_WORD1	104h
EICHD1_WORD1	144h
EICHD2_WORD1	184h
EICHD3_WORD1	1C4h
EICHD4_WORD1	204h

Table continues on the next page...

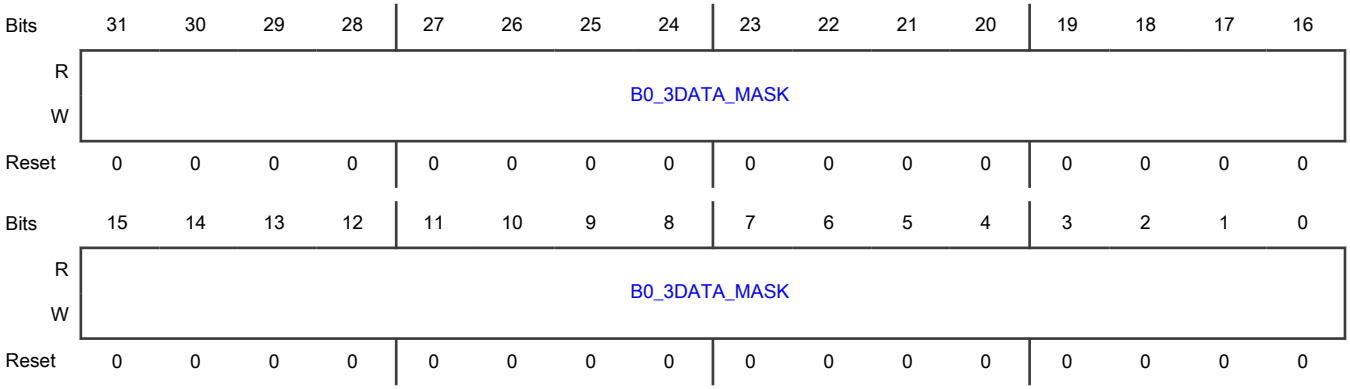
Table continued from the previous page...

Register	Offset
EICHD5_WORD1	244h
EICHD6_WORD1	284h
EICHD7_WORD1	2C4h
EICHD8_WORD1	304h

Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

Diagram



Fields

Field	Function
31-0 B0_3DATA_MASK	Data Mask Bytes 0-3 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored. **NOTE** For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details. 0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified. 1b - The corresponding bit of bytes 0-3 on the read data bus is inverted.

29.6.6 Error Injection Channel Descriptor 7, Word0 (EICHD7_WORD0)

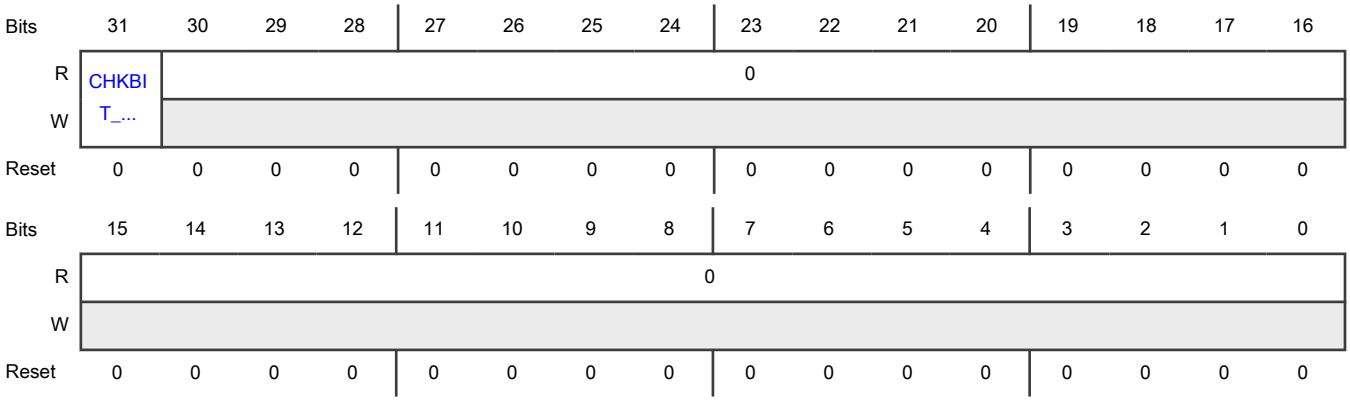
Offset

Register	Offset
EICHD7_WORD0	2C0h

Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

Diagram



Fields

Field	Function
31 CHKBIT_MASK	Checkbit Mask This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored. For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information. **NOTE** Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[0:0] (1 bits wide), CHKBIT_MASK[0] is in the position of the most significant bit. 0b - The corresponding bit of the checkbit bus remains unmodified. 1b - The corresponding bit of the checkbit bus is inverted.
30-0 —	Reserved

29.6.7 Error Injection Channel Descriptor 8, Word0 (EICHD8_WORD0)

Offset

Register	Offset
EICHD8_WORD0	300h

Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CHKBIT_MASK				0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 CHKBIT_MASK	Checkbit Mask This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored. For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information. **NOTE** Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[3:0] (4 bits wide), CHKBIT_MASK[3] is in the position of the most significant bit. 0b - The corresponding bit of the checkbit bus remains unmodified. 1b - The corresponding bit of the checkbit bus is inverted.
27-0 —	Reserved

Chapter 30
Error Recording Module (ERM)

30.1 Chip-specific ERM information

Table 314. Reference links to related information

Topic	Related module	Reference
Full description	ERM	ERM
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

30.1.1 Module instances

This device contains one instance of the Error Recording module, ERM0.

30.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

30.1.3 ERM channel mapping

Table 315. ERM channel mapping

Channel no.	Module	Captured status
0	RAMX	Single-bit error, multiple-bit error, error address, syndrome
1	RAMA	Single-bit error, multiple-bit error, error address, syndrome
2	RAMB	Single-bit error, multiple-bit error, error address, syndrome
3	RAMC	Single-bit error, multiple-bit error, error address, syndrome
4	RAMD	Single-bit error, multiple-bit error, error address, syndrome
5	RAME	Single-bit error, multiple-bit error, error address, syndrome
6	RAMF	Single-bit error, multiple-bit error, error address, syndrome
7	LPCAC RAM	Single-bit error, parity error of each word (syndrome)
8	PKC RAM	Single-bit error, parity error of each byte (syndrome)
9	Flash	ECC double bits error

30.1.4 Channels mapped to non-correctable errors

The following channels are mapped to non-correctable errors in ERM and do not change the count value of the number of correctable ECC error events:

- Channel 7
- Channel 8
- Channel 9

30.2 Overview

The Error Reporting Module (ERM) provides information and optional interrupt notification on memory error events associated with ECC and parity. The ERM collects error events on memory accesses for memory arrays, such as flash memory, system RAM, or peripheral RAMs. ERM supports various channels for memory sources where each ERM channel is associated with a different memory module. See the chip-specific ERM information for details about supported memory sources and specific memory channel assignments. If the memory supports ECC, then ERM syndrome and error address information is captured along with the error event. ERM does not capture syndrome or error address for cache memories or memory with parity instead of ECC.

30.2.1 Features

The ERM includes these features:

- Optional interrupt notification on captured error events
- Capturing of address and syndrome information on single-bit correction and non-correctable ECC events
- Support for error event capturing for memory sources, with individual reporting fields and interrupt configuration per memory channel
- Recording the count value of the number of corrected error events

30.2.2 Block diagram

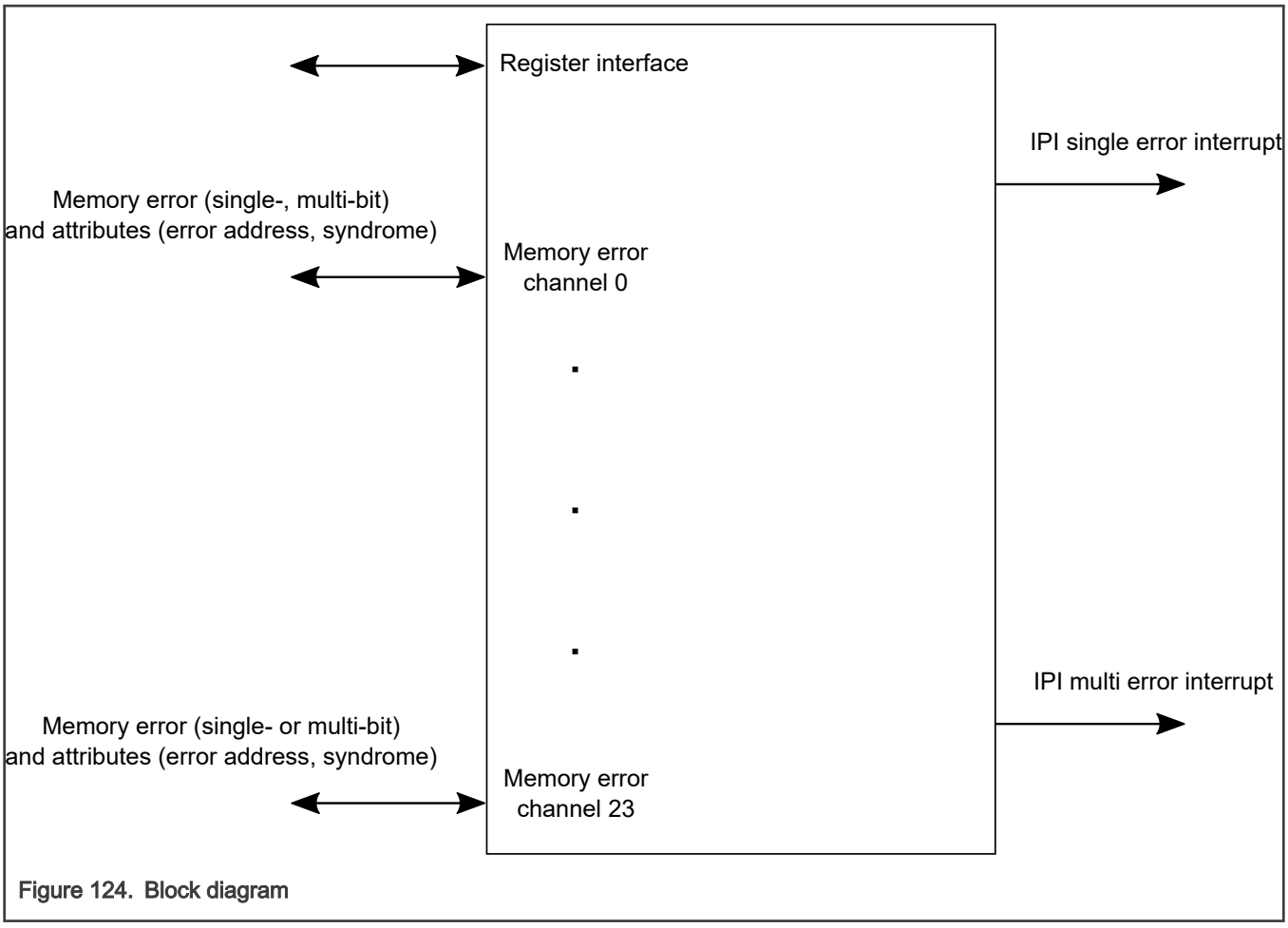


Figure 124. Block diagram

30.3 Functional description

30.3.1 Single-bit correction events

When a single-bit correction event on Memory *n* is detected, the ERM:

- Records the event by changing the value of the applicable Status Register bit SR*x*[SBC*n*] to 1.
- Increments the correctable error count value (until the counter reaches its maximum value): CORR_ERR_CNT*n*[COUNT].
- Records the corresponding access address that initiated the event in the Memory *n* Error Address Register: EAR*n* (if this register is present for the channel).
- Stores the corresponding ECC syndrome in the Memory *n* Error Syndrome Register: SYN*n* (if this register is present for the channel). This register identifies the bit position of the corrected data on single-bit data inversion.

The ERM holds event information only for the last reported event.

To clear the record of an event, write 1 to SR*x*[SBC*n*] to change its value to 0.

To reset the correctable error count value, write all zeros to CORR_ERR_CNT*n*[COUNT].

Optional interrupt notification for single-bit correction events

The ERM provides an option to generate an interrupt notification upon the report of a single-bit correction event. To enable single-bit correction interrupts for a channel:

1. To enable interrupt notification for single-bit correction events on Memory *n*, set CR*x*[ESCIE*n*] to 1.
2. Subsequently, when a single-bit correction event on Memory *n* is detected, the ERM:
   - Records the event and address, and stores the ECC syndrome as usual.
   - Additionally sends an interrupt notification corresponding to the event.
3. To clear both the record of an event and the corresponding interrupt notification, write 1 to SR*x*[SBC*n*] to change its value to 0.

30.3.2 Non-correctable error events

When a non-correctable ECC error event on Memory *n* is detected, the ERM:

- Records the event by changing the value of the applicable Status Register bit: SR*x*[NCE*n*] to 1.
- Records the corresponding access address that initiated the event in the Memory *n* Error Address Register: EAR*n* (if this register is present for the channel).
- Stores the corresponding ECC syndrome in the Memory *n* Error Syndrome Register: SYN*n* (if this register is present for the channel).
  - In the event of a non-correctable address bit inversion, SYN*n* identifies the pertinent address bit position.
  - In the event of a non-correctable, multi-bit data inversion, the syndrome value does not provide any additional diagnostic information.

The ERM holds event information only for the last reported event.

To clear the record of an event, write 1 to SR*x*[NCE*n*] to change its value to 0.

Optional interrupt notification for non-correctable error events

The ERM provides an option to generate an interrupt notification upon the report of a non-correctable ECC event. To enable non-correctable error interrupts for a channel:

1. To enable interrupt notifications for non-correctable error events on Memory *n*, set CR*x*[ENCIE*n*] to 1.
2. Subsequently, when a non-correctable error event on Memory *n* is detected, the ERM:
   - Records the event and address and stores the ECC syndrome as usual.
   - Additionally sends an interrupt notification corresponding to the event.
3. To clear both the record of an event and the corresponding interrupt notification, write 1 to SR*x*[NCE*n*] to change its value to 0.

NOTE

Parity errors can be mapped to non-correctable errors where error attributes like SYNDROME, ADDRESS are not provided.

30.4 Initialization

For each ERM channel supporting memory with ECC, prepare the corresponding memory array before enabling ERM interrupts about errors for that memory.

1. Initialize the memory to a known value so that the correct corresponding ECC codeword is stored.
2. During the memory's initialization, if the ERM captures information about any ECC error event, clear the corresponding SR*x*[SBC*n*] or SR*x*[NCE*n*] field that stores the record of the event.
3. Program the applicable CR*x*[ESCIE*n*] and CR*x*[ENCIE*n*] fields to enable ERM interrupts as desired.

30.5 ERM register descriptions

You can access the programming model:

- Only in supervisor mode
- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an error termination:

- In user mode
- Using non-32-bit access sizes

Based on the design implementation, the following XFR error behavior is evident at the IPS interface.

- Within the ERM memory map, an XFR error is evident at reserved addresses from location 20h to FFh.
- No XFR error is evident at reserved addresses in memory spaces allocated to each channel. For example: For channel 0, for read/write accesses to reserved address 10Ch, the XFR error is 0.
- For accesses to locations beyond the addresses allocated for the final channel, the XFR error is 1.

NOTE

- See the chip-specific ERM information at the beginning of this chapter for details on Memory channel mapping.
- To access the channel registers, corresponding memory channel clock must be enabled.

30.5.1 ERM memory map

ERM0 base address: 4005_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	ERM Configuration Register 0 (CR0)	32	RW	0000_0000h
4h	ERM Configuration Register 1 (CR1)	32	RW	0000_0000h
10h	ERM Status Register 0 (SR0)	32	RW	0000_0000h
14h	ERM Status Register 1 (SR1)	32	RW	0000_0000h
100h	ERM Memory 0 Error Address Register (EAR0)	32	R	0000_0000h
104h	ERM Memory 0 Syndrome Register (SYN0)	32	R	0000_0000h
108h	ERM Memory 0 Correctable Error Count Register (CORR_ERR_CNT0)	32	RW	0000_0000h
110h	ERM Memory 1 Error Address Register (EAR1)	32	R	0000_0000h
114h	ERM Memory 1 Syndrome Register (SYN1)	32	R	0000_0000h
118h	ERM Memory 1 Correctable Error Count Register (CORR_ERR_CNT1)	32	RW	0000_0000h
120h	ERM Memory 2 Error Address Register (EAR2)	32	R	0000_0000h
124h	ERM Memory 2 Syndrome Register (SYN2)	32	R	0000_0000h
128h	ERM Memory 2 Correctable Error Count Register (CORR_ERR_CNT2)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
130h	ERM Memory 3 Error Address Register (EAR3)	32	R	0000_0000h
134h	ERM Memory 3 Syndrome Register (SYN3)	32	R	0000_0000h
138h	ERM Memory 3 Correctable Error Count Register (CORR_ERR_CNT3)	32	RW	0000_0000h
140h	ERM Memory 4 Error Address Register (EAR4)	32	R	0000_0000h
144h	ERM Memory 4 Syndrome Register (SYN4)	32	R	0000_0000h
148h	ERM Memory 4 Correctable Error Count Register (CORR_ERR_CNT4)	32	RW	0000_0000h
150h	ERM Memory 5 Error Address Register (EAR5)	32	R	0000_0000h
154h	ERM Memory 5 Syndrome Register (SYN5)	32	R	0000_0000h
158h	ERM Memory 5 Correctable Error Count Register (CORR_ERR_CNT5)	32	RW	0000_0000h
160h	ERM Memory 6 Error Address Register (EAR6)	32	R	0000_0000h
164h	ERM Memory 6 Syndrome Register (SYN6)	32	R	0000_0000h
168h	ERM Memory 6 Correctable Error Count Register (CORR_ERR_CNT6)	32	RW	0000_0000h
178h	ERM Memory 7 Correctable Error Count Register (CORR_ERR_CNT7)	32	RW	0000_0000h
184h	ERM Memory 8 Syndrome Register (SYN8)	32	R	0000_0000h
188h	ERM Memory 8 Correctable Error Count Register (CORR_ERR_CNT8)	32	RW	0000_0000h
198h	ERM Memory 9 Correctable Error Count Register (CORR_ERR_CNT9)	32	RW	0000_0000h

30.5.2 ERM Configuration Register 0 (CR0)

Offset

Register	Offset
CR0	0h

Function

This 32-bit control register configures the interrupt notification capability for available channels.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ESCIE0	ENCIE0	0		ESCIE1	ENCIE1	0		ESCIE2	ENCIE2	0		ESCIE3	ENCIE3	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ESCIE4	ENCIE4	0		ESCIE5	ENCIE5	0		ESCIE6	ENCIE6	0		ESCIE7	ENCIE7	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 ESCIE0	ESCIE0 Enable Memory 0 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 0 single-bit correction events is disabled. 1b - Interrupt notification of Memory 0 single-bit correction events is enabled.
30 ENCIE0	ENCIE0 Enable Memory 0 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 0 non-correctable error events is disabled. 1b - Interrupt notification of Memory 0 non-correctable error events is enabled.
29-28 —	Reserved
27 ESCIE1	ESCIE1 Enable Memory 1 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 1 single-bit correction events is disabled. 1b - Interrupt notification of Memory 1 single-bit correction events is enabled.
26 ENCIE1	ENCIE1 Enable Memory 1 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 1 non-correctable error events is disabled. 1b - Interrupt notification of Memory 1 non-correctable error events is enabled.
25-24 —	Reserved
23	ESCIE2

Table continues on the next page...

Table continued from the previous page...

Field	Function
ESCIE2	Enable Memory 2 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 2 single-bit correction events is disabled. 1b - Interrupt notification of Memory 2 single-bit correction events is enabled.
22 ENCIE2	ENCIE2 Enable Memory 2 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 2 non-correctable error events is disabled. 1b - Interrupt notification of Memory 2 non-correctable error events is enabled.
21-20 —	Reserved
19 ESCIE3	ESCIE3 Enable Memory 3 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 3 single-bit correction events is disabled. 1b - Interrupt notification of Memory 3 single-bit correction events is enabled.
18 ENCIE3	ENCIE3 Enable Memory 3 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 3 non-correctable error events is disabled. 1b - Interrupt notification of Memory 3 non-correctable error events is enabled.
17-16 —	Reserved
15 ESCIE4	ESCIE4 Enable Memory 4 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 4 single-bit correction events is disabled. 1b - Interrupt notification of Memory 4 single-bit correction events is enabled.
14 ENCIE4	ENCIE4 Enable Memory 4 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 4 non-correctable error events is disabled. 1b - Interrupt notification of Memory 4 non-correctable error events is enabled.
13-12 —	Reserved
11	ESCIE5 Enable Memory 5 Single Correction Interrupt Notification

Table continues on the next page...

Table continued from the previous page...

Field	Function
ESCIE5	0b - Interrupt notification of Memory 5 single-bit correction events is disabled. 1b - Interrupt notification of Memory 5 single-bit correction events is enabled.
10 ENCIE5	ENCIE5 Enable Memory 5 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 5 non-correctable error events is disabled. 1b - Interrupt notification of Memory 5 non-correctable error events is enabled.
9-8 —	Reserved
7 ESCIE6	ESCIE6 Enable Memory 6 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 6 single-bit correction events is disabled. 1b - Interrupt notification of Memory 6 single-bit correction events is enabled.
6 ENCIE6	ENCIE6 Enable Memory 6 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 6 non-correctable error events is disabled. 1b - Interrupt notification of Memory 6 non-correctable error events is enabled.
5-4 —	Reserved
3 ESCIE7	ESCIE7 Enable Memory 7 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 7 single-bit correction events is disabled. 1b - Interrupt notification of Memory 7 single-bit correction events is enabled.
2 ENCIE7	ENCIE7 Enable Memory 7 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 7 non-correctable error events is disabled. 1b - Interrupt notification of Memory 7 non-correctable error events is enabled.
1-0 —	Reserved

30.5.3 ERM Configuration Register 1 (CR1)

Offset

Register	Offset
CR1	4h

Function

This 32-bit control register configures the interrupt notification capability for available channels.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ESCIE8	ENCIE8	0		ESCIE9	ENCIE9	0		Reserved		0		Reserved		0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		0		Reserved		0		Reserved		0		Reserved		0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 ESCIE8	ESCIE8 Enable Memory 8 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 8 single-bit correction events is disabled. 1b - Interrupt notification of Memory 8 single-bit correction events is enabled.
30 ENCIE8	ENCIE8 Enable Memory 8 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 8 non-correctable error events is disabled. 1b - Interrupt notification of Memory 8 non-correctable error events is enabled.
29-28 —	Reserved
27 ESCIE9	ESCIE9 Enable Memory 9 Single Correction Interrupt Notification 0b - Interrupt notification of Memory 9 single-bit correction events is disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Interrupt notification of Memory 9 single-bit correction events is enabled.
26 ENCIE9	ENCIE9 Enable Memory 9 Non-Correctable Interrupt Notification 0b - Interrupt notification of Memory 9 non-correctable error events is disabled. 1b - Interrupt notification of Memory 9 non-correctable error events is enabled.
25-24 —	Reserved
23-22 —	Reserved
21-20 —	Reserved
19-18 —	Reserved
17-16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11-10 —	Reserved
9-8 —	Reserved
7-6 —	Reserved
5-4 —	Reserved
3-2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 —	Reserved

30.5.4 ERM Status Register 0 (SR0)

Offset

Register	Offset
SR0	10h

Function

This 32-bit status register reports error events for available channels.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SBC0	NCE0	0		SBC1	NCE1	0		SBC2	NCE2	0		SBC3	NCE3	0	
W	W1C	W1C			W1C	W1C			W1C	W1C			W1C	W1C		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SBC4	NCE4	0		SBC5	NCE5	0		SBC6	NCE6	0		SBC7	NCE7	0	
W	W1C	W1C			W1C	W1C			W1C	W1C			W1C	W1C		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SBC0	SBC0 Memory 0 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE0] is enabled. 0b - No single-bit correction event on Memory 0 detected. 1b - Single-bit correction event on Memory 0 detected.
30 NCE0	NCE0 Memory 0 Non-Correctable Error Event

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE0] is enabled. 0b - No non-correctable error event on Memory 0 detected. 1b - Non-correctable error event on Memory 0 detected.
29-28 —	Reserved
27 SBC1	SBC1 Memory 1 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE1] is enabled. 0b - No single-bit correction event on Memory 1 detected. 1b - Single-bit correction event on Memory 1 detected.
26 NCE1	NCE1 Memory 1 Non-Correctable Error Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE1] is enabled. 0b - No non-correctable error event on Memory 1 detected. 1b - Non-correctable error event on Memory 1 detected.
25-24 —	Reserved
23 SBC2	SBC2 Memory 2 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE2] is enabled. 0b - No single-bit correction event on Memory 2 detected. 1b - Single-bit correction event on Memory 2 detected.
22 NCE2	NCE2 Memory 2 Non-Correctable Error Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE2] is enabled. 0b - No non-correctable error event on Memory 2 detected. 1b - Non-correctable error event on Memory 2 detected.
21-20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
19 SBC3	SBC3 Memory 3 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE3] is enabled. 0b - No single-bit correction event on Memory 3 detected. 1b - Single-bit correction event on Memory 3 detected.
18 NCE3	NCE3 Memory 3 Non-Correctable Error Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE3] is enabled. 0b - No non-correctable error event on Memory 3 detected. 1b - Non-correctable error event on Memory 3 detected.
17-16 —	Reserved
15 SBC4	SBC4 Memory 4 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE4] is enabled. 0b - No single-bit correction event on Memory 4 detected. 1b - Single-bit correction event on Memory 4 detected.
14 NCE4	NCE4 Memory 4 Non-Correctable Error Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE4] is enabled. 0b - No non-correctable error event on Memory 4 detected. 1b - Non-correctable error event on Memory 4 detected.
13-12 —	Reserved
11 SBC5	SBC5 Memory 5 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE5] is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No single-bit correction event on Memory 5 detected. 1b - Single-bit correction event on Memory 5 detected.
10 NCE5	NCE5 Memory 5 Non-Correctable Error Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE5] is enabled. 0b - No non-correctable error event on Memory 5 detected. 1b - Non-correctable error event on Memory 5 detected.
9-8 —	Reserved
7 SBC6	SBC6 Memory 6 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE6] is enabled. 0b - No single-bit correction event on Memory 6 detected. 1b - Single-bit correction event on Memory 6 detected.
6 NCE6	NCE6 Memory 6 Non-Correctable Error Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE6] is enabled. 0b - No non-correctable error event on Memory 6 detected. 1b - Non-correctable error event on Memory 6 detected.
5-4 —	Reserved
3 SBC7	SBC7 Memory 7 Single-Bit Correction Event Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE7] is enabled. 0b - No single-bit correction event on Memory 7 detected. 1b - Single-bit correction event on Memory 7 detected.
2 NCE7	NCE7 Memory 7 Non-Correctable Error Event

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE7] is enabled. 0b - No non-correctable error event on Memory 7 detected. 1b - Non-correctable error event on Memory 7 detected.
1-0 —	Reserved

30.5.5 ERM Status Register 1 (SR1)

Offset

Register	Offset
SR1	14h

Function

This 32-bit status register reports error events for available channels.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SBC8	NCE8	0		SBC9	NCE9	0		0				0			
W	W1C	W1C			W1C	W1C										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0				0				0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SBC8	SBC8 Memory 8 Single-Bit Correction Event Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE8] is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No single-bit correction event on Memory 8 detected. 1b - Single-bit correction event on Memory 8 detected.
30 NCE8	NCE8 Memory 8 Non-Correctable Error Event Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE8] is enabled. 0b - No non-correctable error event on Memory 8 detected. 1b - Non-correctable error event on Memory 8 detected.
29-28 —	Reserved
27 SBC9	SBC9 Memory 9 Single-Bit Correction Event Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE9] is enabled. 0b - No single-bit correction event on Memory 9 detected. 1b - Single-bit correction event on Memory 9 detected.
26 NCE9	NCE9 Memory 9 Non-Correctable Error Event Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE9] is enabled. 0b - No non-correctable error event on Memory 9 detected. 1b - Non-correctable error event on Memory 9 detected.
25-24 —	Reserved
23-20 —	Reserved
19-16 —	Reserved
15-12 —	Reserved
11-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-4 —	Reserved
3-0 —	Reserved

30.5.6 ERM Memory a Error Address Register (EAR0 - EAR6)

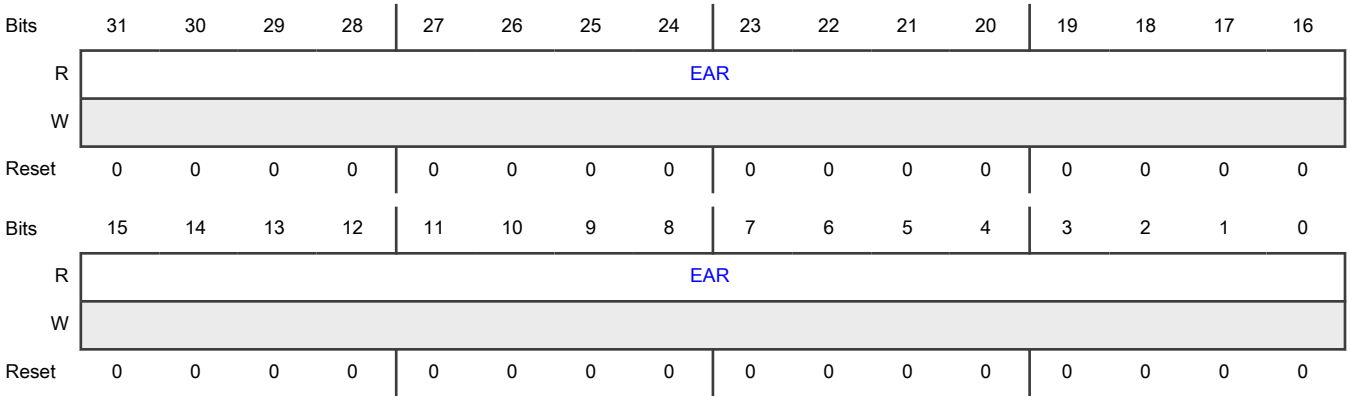
Offset

Register	Offset
EAR0	100h
EAR1	110h
EAR2	120h
EAR3	130h
EAR4	140h
EAR5	150h
EAR6	160h

Function

Each ERM Memory n Error Address Register is a 32-bit register for capturing the address of the last ECC event in Memory*n*, where*n* denotes the memory channel. Any attempted write to EAR*n* is ignored.

Diagram



Fields

Field	Function
31-0 EAR	EAR Memory*n* Error Address — This field contains the faulting system address of the last recorded ECC event on Memory*n*.

30.5.7 ERM Memory a Syndrome Register (SYN0 - SYN8)

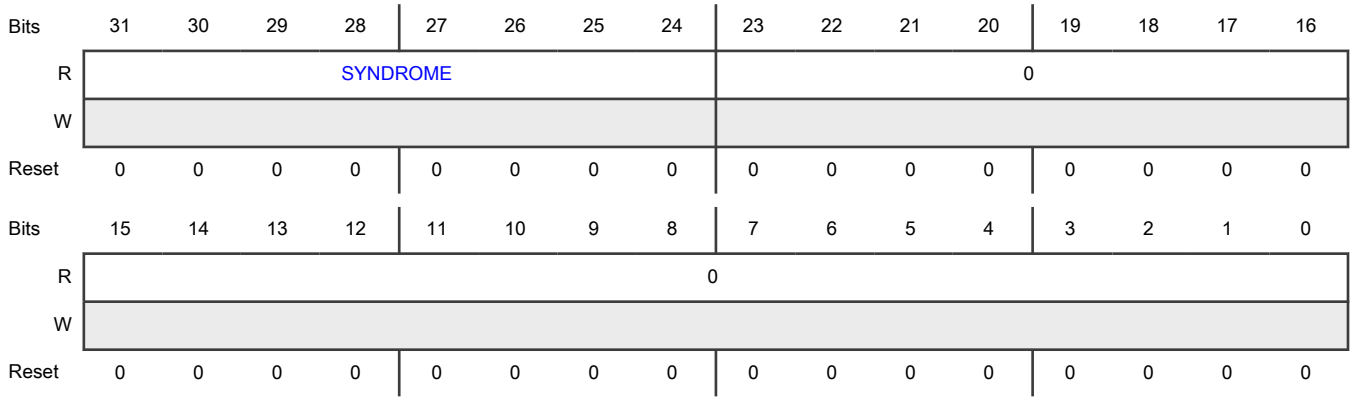
Offset

Register	Offset
SYN0	104h
SYN1	114h
SYN2	124h
SYN3	134h
SYN4	144h
SYN5	154h
SYN6	164h
SYN8	184h

Function

The ERM Memory *n* Syndrome Register is a 32-bit register for capturing the calculated syndrome of the last ECC event on Memory*n*, where*n* denotes the memory channel. Any attempted write to SYN*n* is ignored. The syndrome value identifies the pertinent bit position on a correctable, single-bit data inversion or a non-correctable, single-bit address inversion. The syndrome value does not provide any additional diagnostic information on non-correctable, multi-bit inversions.

Diagram



Fields

Field	Function
31-24 SYNDROME	SYNDROME Memory*n* Syndrome — This field contains the ECC syndrome associated with the last recorded ECC event on Memory*n*.
23-0 —	Reserved

30.5.8 ERM Memory a Correctable Error Count Register (CORR_ERR_CNT0 - CORR_ERR_CNT9)

Offset

Register	Offset
CORR_ERR_CNT0	108h
CORR_ERR_CNT1	118h
CORR_ERR_CNT2	128h
CORR_ERR_CNT3	138h
CORR_ERR_CNT4	148h
CORR_ERR_CNT5	158h
CORR_ERR_CNT6	168h
CORR_ERR_CNT7	178h
CORR_ERR_CNT8	188h
CORR_ERR_CNT9	198h

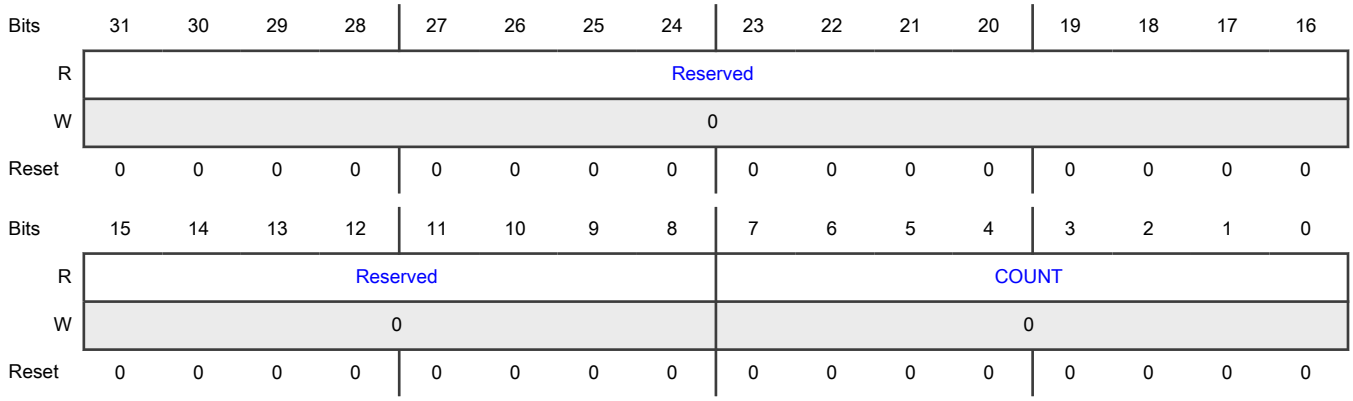
Function

Each 32-bit ERM Memory *n* Correctable Error Count Register records the count value of the number of correctable ECC error events for Memory*n*, where*n* denotes the memory channel.

NOTE

Non-correctable errors are considered a serious fault, so the ERM does not provide any mechanism to count non-correctable errors. Only correctable errors are counted.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 COUNT	Memory n Correctable Error Count For each correctable error event, the ERM increments this field's error count value until the counter reaches its maximum value FFh. COUNT value will stop when it reaches maximum value FFh and will not wrap even if additional errors occur. Read this field to determine the correctable error count value so far. Write all zeros to this field to reset the counter. Writing a non-zero value has no effect.

Chapter 31
Programmable Logic Unit (PLU)

31.1 Chip-specific PLU information

Table 316. Reference links to related information

Topic	Related module	Reference
Full description	PLU	PLU
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

31.1.1 Module instances

This device has one instance of PLU module, PLU0.

31.1.2 PLU configuration

PLU is configurable along two parameters: number of LUT elements and number of primary inputs. The chip uses the following configuration:

- 26 LUT elements
- 6 Primary inputs (does not include the clock input, use of which is optional)
- 8 Primary outputs

31.2 Overview

The Programmable Logic Unit (PLU) is used to create both small combinatorial and sequential logic networks including simple state machines. PLU continuously monitors the state of input devices and uses a custom program to control the state of output devices.

31.2.1 Features

PLU includes the following features:

- An array of 26 inter-connectable, 5-input look-up table (LUT) elements, and four flip-flops.
  - Each LUT element has 5 input multiplexer registers and one truth table (look-up table) register.
  - Each LUT input has 35 possible input sources to choose from.
  - All inputs connected directly to the package pins via chip-level I/O multiplexing.
- Eight primary output pins, selected using a multiplexer from among the 26 LUT outputs and the four flip-flops.
  - Up to four of the LUT outputs can be captured in one of the four state flip-flops. They can either be connected to the inputs of other LUTs or be used as primary outputs, or they can be used in both of those functions simultaneously.
- An external clock to drive the four flip-flops if a sequential network is implemented.
- Programmable logic that can drive on-chip inputs or triggers through external pin-to-pin connections.
- A tool suite to facilitate programming of the PLU to implement the logic network.

31.2.2 Block diagram

The following figure shows the PLU block diagram with the following configuration: 26, 5-input LUT elements.

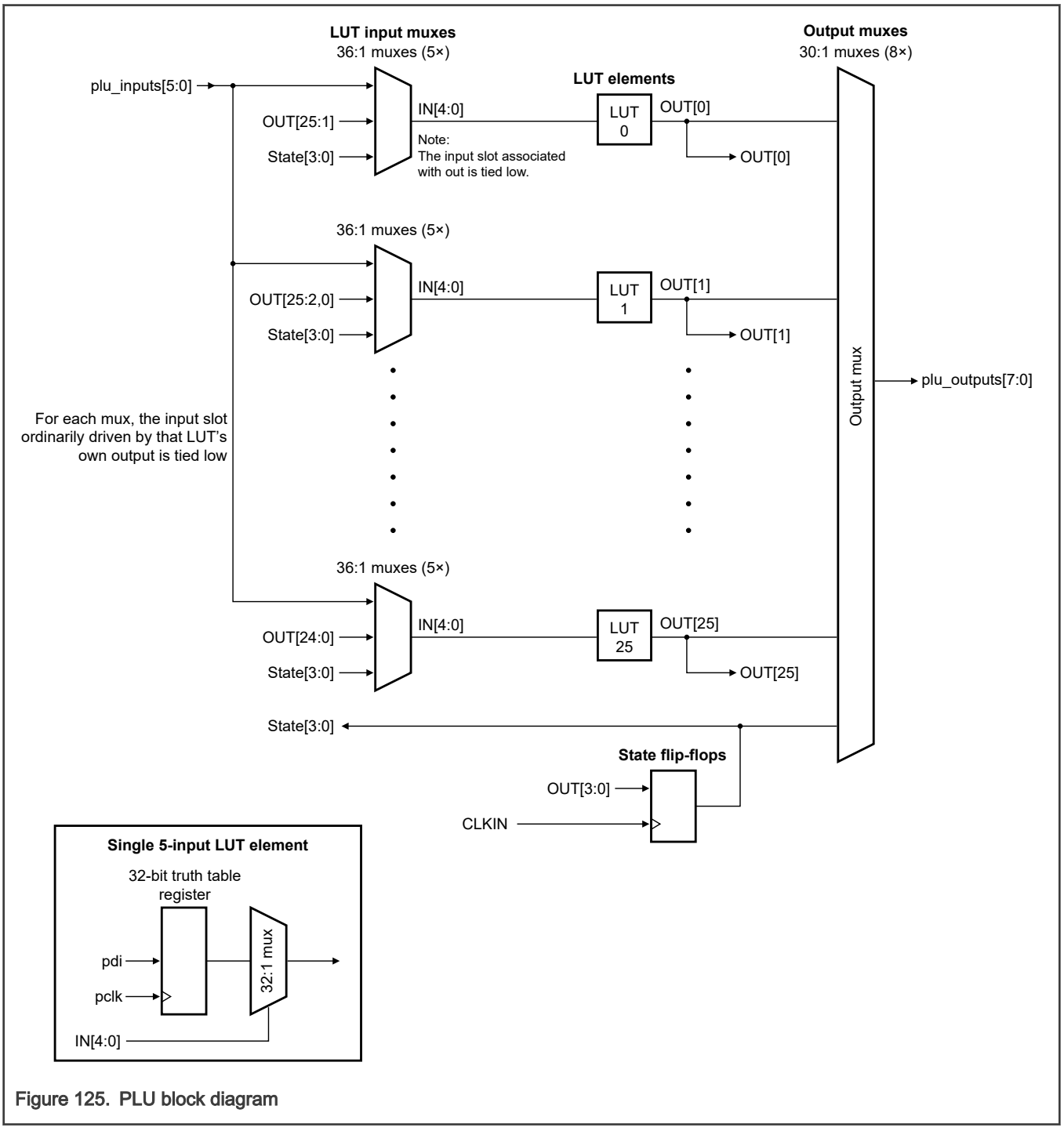


Figure 125. PLU block diagram

31.3 Functional description

PLU is comprised of twenty-six 5-input LUT elements. Each LUT element contains a 32-bit truth table (look-up table) register and a 32:1 multiplexer. During operation, the 5 LUT inputs control the select lines of the multiplexer. This structure allows you to use any logical combination of the 5 LUT inputs.

The 5 inputs to each LUT can be driven from a selection of sources comprised of the primary inputs from pins, the outputs of the other LUTs, and the outputs of the four state flip-flops.

A set of multiplexers associated with each LUT is used to select the 5 inputs to that LUT. These multiplexers are controlled by registers programmed during initialization. Connecting multiple LUT elements together permits construction of complex boolean expressions.

Outputs of up to four of the LUTs can be captured in the four state flip-flops and can be used as primary outputs or connected to the inputs of other LUTs. These four flip-flops, if used by the target logic network, are clocked by the external CLKIN supplied by the user.

The four state flip-flops are automatically cleared to zero by the internal chip reset signal. If a different initial state than all zeros is required for these flip-flops, you must force the primary inputs at startup to a combination that will achieve the required state. That startup combination must be maintained through at least one CLKIN rising-edge.

Generally, the PLU bus clock can be shut off to conserve power after the PLU module is configured. If you need to read the outputs register when PLU is operational, the PLU bus clock must be re-enabled prior to performing the read.

31.3.1 Clocking

The clock that feeds the programmable logic unit can be selected from the following clock inputs:

Global Functional Clock - pclk

- This clock is supposed to be on in normal operations, all of the registers shown in capture 5 are clocked by the internal bus clock.

External Clock - plu_clkin

- This is the external clock from outside the chip which can be used to run the four state flip-flops. Only the four state flip-flops used as part of the target logic network use the externally applied plu_clkin.

Low Reference Clock - hclk

- This clock is used for programmable glitch filter, when no plu_clkin is present, an alternative mechanism for suppressing glitches is provided by means of a programmable, digital glitch filter. hclk is used during deep-sleep mode if required.It will be a low-frequency, low-power clock (For example, a 1 MHz low-power oscillator).

High frequency Clock - hclk_div_2

- This clock performs the same function as hclk, but at a different frequency. hclk_div_2 will be a higher frequency clock (For example,a 12 MHz 16 MHz or 30 MHz FRO) which will be higher-power but provide shorter latency and finer resolution over the filter width.

Medium frequency Clock - hclk_div_4

- This clock performs the same function as hclk or hclk_div_2, but at a different frequency. Should be tied-low and de-spec'd for most chips. Faster or slower clocks 1MHz - 30 MHz not useful.

31.3.2 Programming PLU

Programming PLU to implement a particular logic network requires the following steps:

1. Write to each truth table register to specify the logic functions to be performed by each of the LUT elements.
2. Program the input multiplexer registers to select the five inputs presented to each LUT.
3. Program the output multiplexer register to select the eight primary outputs from the PLU module.

NOTE

Programming of all registers must be performed during initialization. Do not write to the registers during operation.

A tool suite is provided to facilitate the programming of PLU. The tool suite reports the values that must be written to all the registers in order to implement a logic network. See PLU programming tool suite for a complete description of the tool flow.

The I/O multiplexing must be programmed in order to connect the required number of PLU primary inputs and outputs to pins.

31.3.2.1 PLU programming tool suite

The tool suite that is provided facilitates configuring PLU in order to implement the desired logic network. See the tool suite documentation for input information.

The output from the tool suite includes the following:

- Values to be programmed into all LUT*x*_INPUT*x*_MUX registers.
- Values to be programmed into all LUT*x*_TRUTH registers.
- Values to be programmed into all OUTPUT*x*_MUX registers.
- Error response if the described network cannot be implemented—most commonly due to an excessive amount of logic or the use of more than four flip-flops.
- AC Timing parameters for the implemented design.

31.3.3 Interrupts

The raw wake-up or interrupt output generated by PLU is prone to glitching because of the frequent delays within the network of LUTs. If used directly, the raw wake-up or interrupt output is likely to generate false wake-ups and interrupt requests. Two options for eliminating glitches are provided in the following topics.

NOTE

Glitch pulse-widths can reach lengths in excess of tens of nanoseconds.

31.4 External signals

A particular logic network may not require all of the available inputs or outputs. You can specify which inputs and outputs to use and which package pins those inputs and outputs will connect to as part of the overall top-level I/O configuration.

If the logic network uses one or more of the four state flip-flops, an external clock must be applied to the CLKIN input. The package pin used for this function is specified using the chip-level I/O multiplexing. All other PLU inputs must meet specified setup and hold times relative to this clock input. Output timing is also specified relative to this pin.

NOTE

If the logic network is purely combinatorial, there is no need to provide an input clock to CLKIN.

Table 317. External signals

Signal	I/O	Description
plu_inputs[5:0]	Input	Primary inputs. All plu_inputs are available as input sources to all LUT elements.
plu_clkin	Input	Input clock to the four state flip-flops, if used. Not required for purely combinatorial networks. Input/output timing is specified relative to this clock.
plu_outputs[7:0]	Input/Output	Primary outputs. Selectable via multiplexers from among all LUT element outputs and the four state flip-flops.

31.5 Initialization

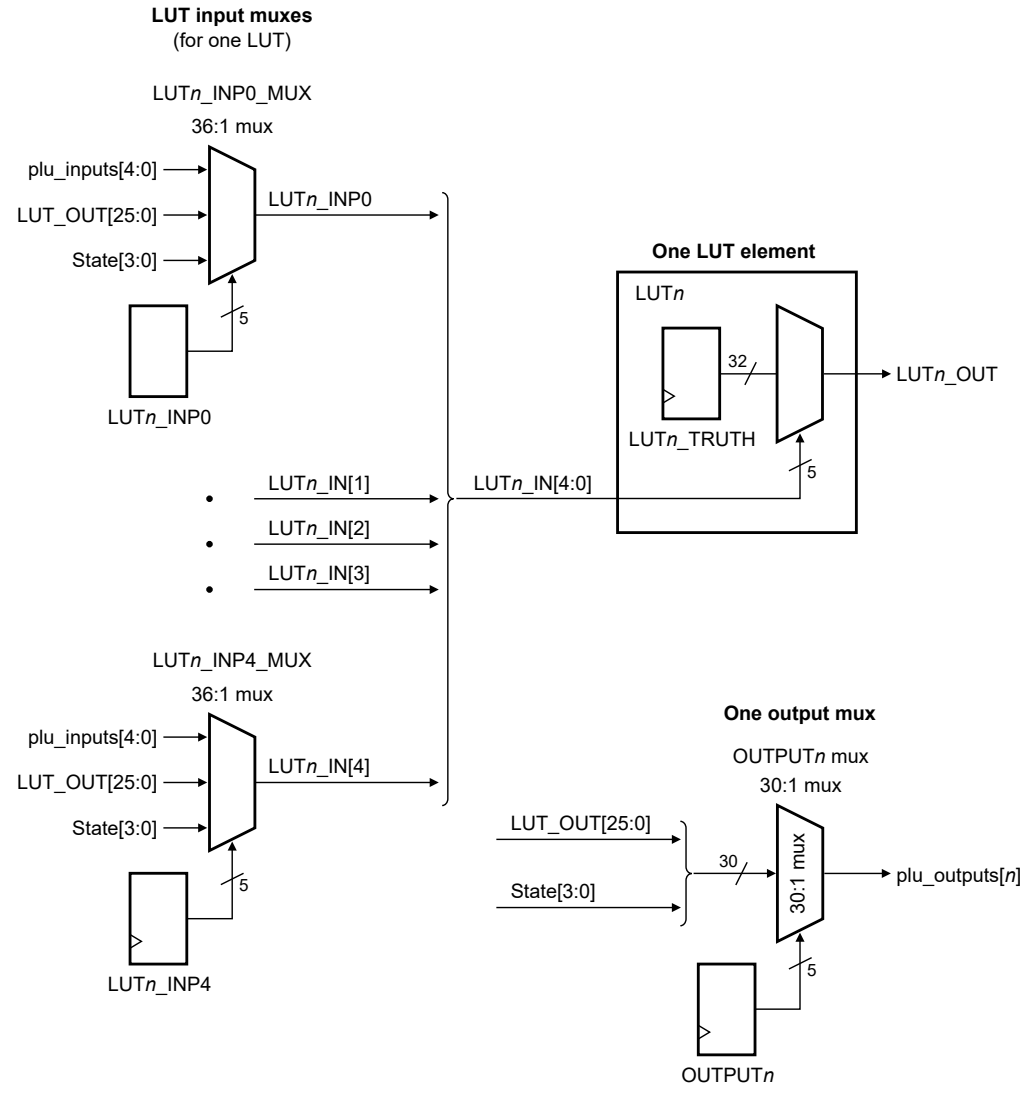
To initialize PLU:

- Enable the clock to the PLU register interface.
- Enable PLU interrupts for waking up from deep-sleep and low-power modes.
- Configure the various truth table registers, input multiplexer registers, output multiplexer register.

31.6 Memory Map and register definition

All of the registers shown are clocked by the internal bus clock. Only the four state flip-flops used as part of the target logic network use the externally applied CLKIN.

All ones in the implemented bits (lsbs) for the LUT*n*_INP*x* and OUTPUT mux registers means none of the input options are selected. In this case, the associated multiplexer output is a fixed logic '0'. These registers must be programmed to a valid value.



Note: For each mux, the input slot driven by the LUT's own output is tied low.

Figure 126. PLU register multiplexing configuration

31.6.1 Programmable Logic Unit (PLU) register descriptions

31.6.1.1 PLU memory map

PLU0 base address: 4003_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 330h	Input select register for LUTn (0 to 25), Inputx (5 inputs) (LUT0_INP0_MUX - LUT25_INP4_MUX)	32	RW	See section
800h - 864h	PLU LUT truth table (LUT0_TRUTH - LUT25_TRUTH)	32	RW	0000_0000h
900h	PLU outputs (OUTPUTS)	32	R	0000_0000h
904h	Wakeup interrupt control (WAKEINT_CTRL)	32	RW	0000_0000h
C00h - C1Ch	PLU output multiplexer (OUTPUT0_MUX - OUTPUT7_MUX)	32	RW	0000_001Fh

31.6.1.2 Input select register for LUTn (0 to 25), Inputx (5 inputs) (LUT0_INP0_MUX - LUT25_INP4_MUX)

Offset

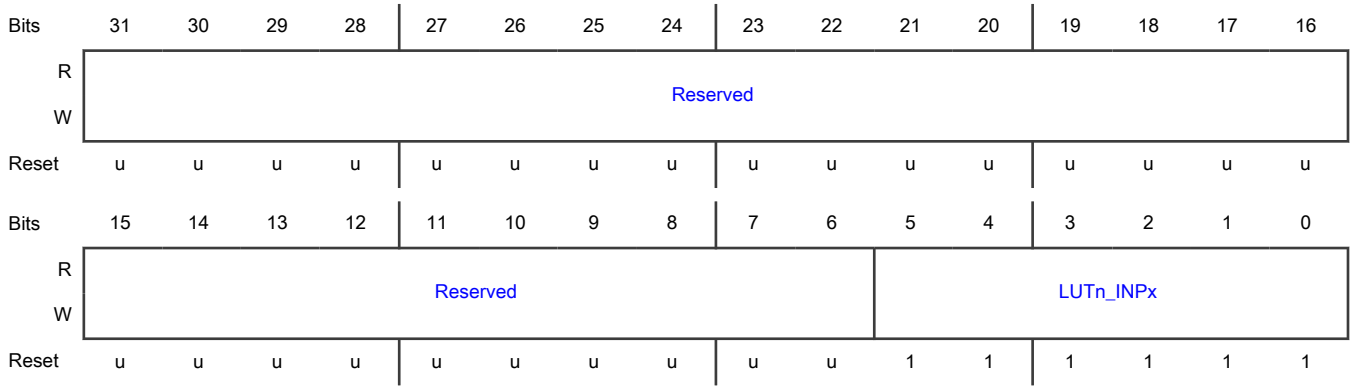
For a = 0 to 25; b = 0 to 4:

Register	Offset
LUTa_INPb_MUX	0h + (a × 20h) + (b × 4h)

Function

These registers control the multiplexers to select which input sources to connect to each LUT input.

Diagram



Fields

Field	Function
31-6 —	Reserved Software should not write ones to reserved bits.
5-0 LUTn_INPx	Selects the input source to be connected to LUTn_INPx For each LUT, the slot associated with the output from LUTn itself is tied low.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The available input sources for each LUT are, in order: 1. PLU primary inputs, beginning with plu_inputs[0]. 2. Outputs from all of the other LUT elements (aside from LUTn itself) in order from the lowest to highest-numbered remaining LUTs. For each LUT, the slot associated with the output from LUTn itself is tied low. 3. Four state flip-flops, beginning with state[0]. **NOTE** The reset value of all ones causes a fixed '0' to be passed to this LUT input. 111111b - Fixed 0 00_0000b - PLU primary inputs 0 00_0001b - PLU primary inputs 1 00_0010b - PLU primary inputs 2 00_0011b - PLU primary inputs 3 00_0100b - PLU primary inputs 4 00_0101b - PLU primary inputs 5 00_0110b - Output of LUT0 00_0111b - Output of LUT1 00_1000b - Output of LUT2 00_1001b - Output of LUT3 00_1010b - Output of LUT4 00_1011b - Output of LUT5 00_1100b - Output of LUT6 00_1101b - Output of LUT7 00_1110b - Output of LUT8 00_1111b - Output of LUT9 01_0000b - Output of LUT10 01_0001b - Output of LUT11 01_0010b - Output of LUT12 01_0011b - Output of LUT13 01_0100b - Output of LUT14 01_0101b - Output of LUT15 01_0110b - Output of LUT16 01_0111b - Output of LUT17

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1000b - Output of LUT18 01_1001b - Output of LUT19 01_1010b - Output of LUT20 01_1011b - Output of LUT21 01_1100b - Output of LUT22 01_1101b - Output of LUT23 01_1110b - Output of LUT24 01_1111b - Output of LUT25 10_0000b - State[0] 10_0001b - State[1] 10_0010b - State[2] 10_0011b - State[3]

31.6.1.3 PLU LUT truth table (LUT0_TRUTH - LUT25_TRUTH)

Offset

For a = 0 to 25:

Register	Offset
LUTa_TRUTH	800h + (a × 4h)

Function

Each LUT element contains one 32-bit truth table register which specifies whether the LUT will output a '0' or a '1' for each combination of the LUT inputs. In other words, these registers specify the complex Boolean expression each individual LUT is to perform.

NOTE

The values that must be programmed into each of these registers is provided by the tool suite.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LUT_TRUTH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LUT_TRUTH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 LUT_TRUTH	LUT truth table Specifies the truth table contents.

31.6.1.4 PLU outputs (OUTPUTS)

Offset

Register	Offset
OUTPUTS	900h

Function

This register provides the current state of the eight designated PLU outputs. Due to the asynchronous nature of PLU, there is no guarantee that a read of this register will not capture transitional data. It is strongly recommended to read this data multiple times until a consistent result is returned, unless it is known that the PLU inputs will be stable during the read operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								OUTPUT_STATE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved Software should not write ones to reserved bits.
7-0 OUTPUT_STATE	Output state Provides the current state of the eight designated PLU outputs and the four state flip-flop outputs.

31.6.1.5 Wakeup interrupt control (WAKEINT_CTRL)

Offset

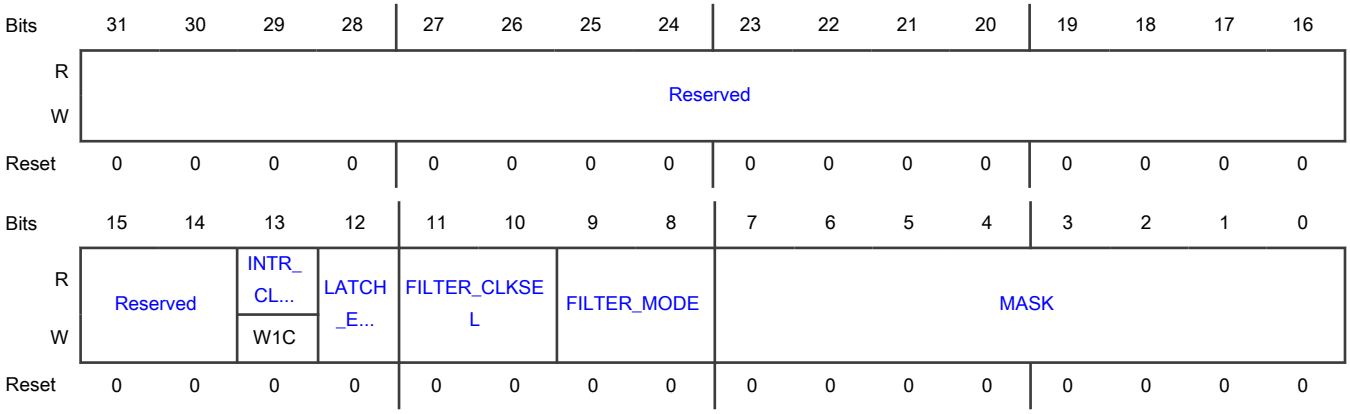
Register	Offset
WAKEINT_CTRL	904h

Function

This register enables PLU outputs to contribute to an asynchronous wake-up or interrupt request. All enabled output signals are logically combined using OR to generate a single wake-up or interrupt request.

A '1' in each bit in this register enables the corresponding PLU output to contribute to wake-up or interrupt generation. All enabled PLU outputs are combined using OR to generate the wake-up or interrupt request. A '0' in any bit of this register blocks the corresponding PLU output from causing a wake-up or interrupt. See Interrupts for more information.

Diagram



Fields

Field	Function
31-14 —	Reserved Software should not write ones to reserved bits.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 INTR_CLEAR	Write to clear wakeint_latched When using the registered wake-up or interrupt option (LATCH_ENABLE = '1') writing a '1' to this bit will clear the wake-up or interrupt request flag. Writing a '0' to this bit has no effect. There will be a delay of up to 1.5 CLKIN clock cycles before this write-to-clear takes effect, at which point this bit will also be cleared. Software can poll this bit to determine when the wake-up or interrupt request has been removed. **NOTE** It is not necessary for the PLU bus clock to be enabled in order to write-to or read-back this bit. This is not the case for the other bits of this register.
12 LATCH_ENABLE	Latch the interrupt Latch the interrupt and then the interrupt can be cleared with the next bit INTR_CLEAR. Setting this bit specifies use of the registered version of the wake-up or interrupt request instead of the raw or glitch-filtered version. This option can only be used if a CLKIN clock is provided from off-chip. If this bit is set, the wake-up or interrupt request will be set on the rising-edge of CLKIN whenever the raw wake-up or interrupt signal is high. The request must be cleared by software. **NOTE** This mode is not compatible with use of the glitch filter. If this bit is set, the FILTER_MODE field should be set to '00' (Bypass Mode).
11-10 FILTER_CLKSEL	Filter clock select Selects which clock source option to use as the filter clock. See the chip-specific PLU information for the clocking options. 00b - Selects the 1 MHz low-power oscillator as the filter clock. 01b - Selects the 12 MHz FRO as the filter clock. 10b - Reserved 11b - Reserved
9-8 FILTER_MODE	Filter Mode Control input of the PLU, add filtering for glitch. 00b - Bypass mode. Filtering is disabled. The raw wake-up or interrupt will be passed through. 01b - Filter 1 clock period. Any pulse duration shorter than one cycle of the designated filter clock will be filtered out. Pulse widths up to two cycles long may be filtered. 10b - Filter 2 clock period. Any pulse duration shorter than two cycles of the designated filter clock will be filtered out. Pulse widths up to three cycles long may be filtered. 11b - Filter 3 clock period. Any pulse duration shorter than three cycles of the designated filter clock will be filtered out. Pulse widths up to four cycles long may be filtered.
7-0 MASK	Interrupt mask Selects which of the eight PLU outputs contribute to interrupt.

31.6.1.6 PLU output multiplexer (OUTPUT0_MUX - OUTPUT7_MUX)

Offset

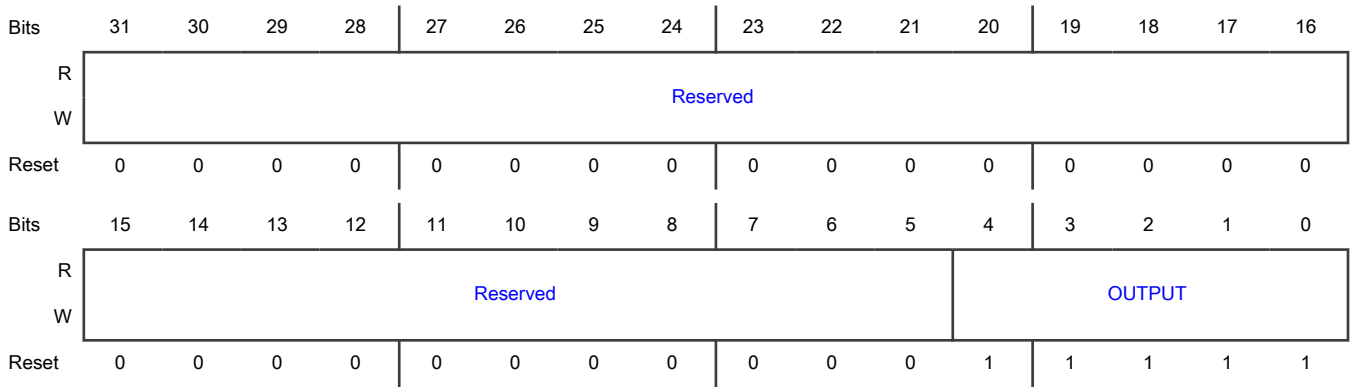
For a = 0 to 7:

Register	Offset
OUTPUTa_MUX	C00h + (a × 4h)

Function

Selects the source to be connected to PLU output *n*. The available choices are all of the individual LUT element outputs and the four state flip-flop outputs.

Diagram



Fields

Field	Function
31-5 —	Reserved Software should not write ones.
4-0 OUTPUT	Selects the source to be connected to PLU output n. For each PLU output the available sources are, in order: 1. The outputs from all of the available LUT elements, beginning with LUT0. 2. The four state flip-flops, beginning with state[0]. 0x0 programmed selects LUT0 as the source of this output. Each higher binary value selects one of the other sources mentioned above. 11111b - Fixed 0 0_0000b - LUT output 0 0_0001b - LUT output 1 0_0010b - LUT output 2 0_0011b - LUT output 3

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0100b - LUT output 4 0_0101b - LUT output 5 0_0110b - LUT output 6 0_0111b - LUT output 7 0_1000b - LUT output 8 0_1001b - LUT output 9 0_1010b - LUT output 10 0_1011b - LUT output 11 0_1100b - LUT output 12 0_1101b - LUT output 13 0_1110b - LUT output 14 0_1111b - LUT output 15 1_0000b - LUT output 16 1_0001b - LUT output 17 1_0010b - LUT output 18 1_0011b - LUT output 19 1_0100b - LUT output 20 1_0101b - LUT output 21 1_0110b - LUT output 22 1_0111b - LUT output 23 1_1000b - LUT output 24 1_1001b - LUT output 25 1_1010b - State[0] 1_1011b - State[1] 1_1100b - State[2] 1_1101b - State[3]

Chapter 32
Reset

32.1 Overview

The following table shows the reset sources for this chip. Each system reset source has a corresponding field maintained in the system reset status (SRS) register through reset. Use this information to take appropriate action when coming out of a reset.

Table 318. Reset sources

Reset type	Source
Power	• Power-on reset (POR) • Low-voltage detect (LVD) • High-voltage detect (HVD)
External	• External pin reset (RESET_b) • WUU
Internal	• VBAT domain POR • Code watchdog reset • Intrusion and tamper response controller (ITRC) reset • Tamper event reset • WWDT • Software (SW) • SCG • Reset acknowledge timeout (RSTACK) • Low-power acknowledge timeout (LPACK) • Lock up (LOCKUP) • Debug port (DAP) • JTAG

32.2 Power reset sources

Supply voltages must remain in their specified ranges during execution to ensure the predictable operation of the chip. Because this is not possible during power up and power down, the chip implements multiple voltage monitors to detect when the supply is outside an acceptable range. When this occurs, the voltage monitor initiates a reset to halt execution and prevent unexpected behavior.

The voltage monitors detect power-on, low-voltage, and high-voltage events.

32.2.1 Power-on reset (POR)

The POR voltage monitor ensures that the voltage applied to the system is high enough for other analog modules (such as bias circuits and low-voltage monitors) to operate correctly.

When you initially apply voltage to the chip or when the supply voltage drops below the POR falling threshold, the POR monitor triggers the POR condition to the chip.

The POR condition asserts a POR to all power domains. A POR sets the appropriate fields in the CMC registers so that software can determine when a POR occurs. See POR block diagram for more information.

32.2.2 Low-voltage detect (LVD)

The LVD monitors ensure that the voltage supplies to the chip are high enough for the logic and memories to operate correctly. When voltage is applied initially to the chip, the POR voltage monitor holds the LVD monitors in reset until the voltages are high enough. The LVD monitors hold the chip in reset until all monitoring voltages have risen above the respective default LVD low thresholds. The LVD monitors are enabled by default.

When enabled, the LVD monitor triggers a reset when the voltage drops below the V_{LVD} threshold. The voltage threshold that triggers the LVD reset is programmable. The LVD monitor ensures that the chip halts operation and remains in reset when the monitored voltage is outside the desired range.

In active modes, the LVD monitors are enabled by default. They are optionally available in certain low-power modes. See Module operation in low-power modes for more information on the available modes.

The reset sources, which the individual LVD monitors trigger, combine into a single LVD reset. The LVD reset condition initiates a POR in all power domains while only the POR, LVD, and HVD detection logic remain operational. An LVD reset sets the appropriate fields in the CMC registers so that software can determine when an LVD reset occurs.

32.2.3 High-voltage detect (HVD)

The HVD monitors ensure that the voltage supplies to the chip are within the operating range for the logic and memories using those power supplies. The HVD monitors are disabled by default, but when an HVD circuit is enabled, the HVD monitor triggers a reset when the voltage rises above the V_{HVD} threshold. The HVD monitor ensures that the chip halts operation and remains in reset when the monitored voltage is outside the desired range.

The HVD monitors are available in Active and certain low-power modes. See Module operation in low-power modes for more information on the available modes.

The reset sources, which the individual HVD monitors trigger, combine into a single HVD reset. The HVD reset condition initiates a POR in all power domains while only the POR, LVD, and HVD detection logic remain operational. An HVD reset sets the appropriate fields in the CMC registers so that software can determine when an HVD reset occurs.

32.3 External reset sources

The chip supports external resets to reset or awaken the chip from very low-power states. This allows the chip to start at the correct time from a known state.

32.3.1 External pin reset (RESET_b)

The RESET_b pin is a bidirectional, open-drain pin with an internal pullup resistor. The RESET_b pin is functional in two modes:

- During Active and low-power modes, you can assert the RESET_b pin externally to force the chip into a pin-reset condition.
- During reset, RESET_b remains asserted until the chip completes hardware initialization, at which point the RESET_b pin is released. If you assert the RESET_b pin externally, the chip remains in reset until you deassert RESET_b.

The RESET_b signals passed through two digital filters. CMC's reset pin control register configures the RESET_b signals. In Active mode, you can use the CMC reset filter and configure the duration of the filter from 1 to 32 CMC clock cycles. The CMC reset filter clock source is slow_clk. In both Low-Power mode and Active mode, you can enable a low-power reset filter to filter RESET_b signals for the duration of two clock cycles. The clock is sourced from either FRO_16K or OSC_32K. Use RTC.CTRL[CLK_SEL] to select OSC_32K as the lp_osc clock. See Table 319 for more information.

The pin reset condition triggers CMC to assert a warm reset signal to all power domains. A pin reset sets the appropriate fields in the CMC registers so that software can determine when a pin reset occurs.

32.3.2 WUU

WUU provides a number of external pins and on-chip peripherals to wake the chip from Deep Power-down mode. See the WUU chapter for a list of external and internal reset sources connected to WUU.

After wake-up from Deep Power-down mode, the power management logic triggers a wake-up reset in the power domains that were previously powered off. The wake-up reset condition triggers CMC to assert a POR signal to the power domains that were powered down. Wake-up reset does not affect the system voltage domain. A wake-up reset sets the appropriate fields in the CMC registers so that software can determine when a wake-up reset occurs.

32.4 Internal reset sources

When certain erroneous conditions are detected, the chip resets itself internally. The internal reset allows the chip to recover from the erroneous conditions and restart from a known state.

32.4.1 VBAT domain POR

The chip resets when the VBAT domain POR asserts.

32.4.2 Code watchdog reset

The code watchdog timer (CDOG) module detects unexpected changes (faults) in the code execution flow to protect the software integrity. You can configure CDOG to reset or interrupt the processor core when the module detects a fault.

32.4.3 Intrusion and tamper response controller (ITRC) reset

System security violation (SECVIO) events connect to ITRC inputs. You can configure ITRC to generate a reset when any ITRC input is asserted.

32.4.4 Tamper event reset

The Digital Tamper (TDET) module supports active and passive tamper pin function. When a pin tamper event is detected, TDET can generate a reset.

32.4.5 WWDT

One of the WWDT modules generates the WWDT reset.

WWDT receives periodic communication from software, known as servicing or refreshing WWDT, to monitor the operation of the system. If periodic servicing does not occur, the watchdog triggers a WWDT reset condition.

The WWDT reset condition triggers CMC to assert a warm reset signal to all power domains. A WWDT reset sets the appropriate fields in the CMC registers so that software can determine when a WWDT reset occurs.

32.4.6 Software (SW)

During execution, software can initiate a reset to restart the chip from a known state if it detects an erroneous operation.

The SW reset condition, which CPU0 SYSRESETREQ generates, triggers CMC to assert a warm reset signal to all power domains. An SW reset sets the appropriate fields in the CMC registers so that software can determine when an SW reset occurs. See the system reset status register (SRS) in the CMC chapter for more information.

32.4.7 SCG

SCG contains a clock monitor, which detects a loss of the external clock when enabled. A loss of the external clock triggers the SCG reset condition.

The SCG reset condition triggers a warm reset in all power domains. When CMC.SRS[SCG] and CMC.SRS[WARM] status fields are 1, an SCG reset condition has occurred.

See the SCG chapter for more information on enabling the clock monitor.

NOTE

To prevent unexpected reset events, you must disable clock monitors before entering any low-power modes where the external clocks are absent.

32.4.8 Reset acknowledge timeout (RSTACK)

The reset state machine includes a timeout counter. If the reset state machine does not progress within 64 ms, it triggers the timeout counter and the RSTACK reset condition.

The RSTACK reset condition triggers CMC to assert a warm reset signal to all power domains. An RSTACK reset sets the appropriate fields in the CMC registers so that software can determine when an RSTACK reset occurs.

32.4.9 Low-power acknowledge timeout (LPACK)

The low-power entry state machine includes a timeout counter that is triggered if a module does not acknowledge entry into a low-power mode after 64 ms. This triggers the LPACK reset condition.

The LPACK reset condition triggers CMC to assert a warm reset signal to all power domains. An LPACK reset sets the appropriate fields in the CMC registers so that software can determine when an LPACK reset occurs.

32.4.10 Lock up (LOCKUP)

The Cortex-M33 core enters the lockup state as a result of certain illegal operations. This triggers the lockup reset condition.

The lockup reset condition triggers CMC to assert a warm reset signal to all power domains. A lockup reset sets the appropriate fields in the CMC registers so that software can determine when a lockup reset occurs.

32.4.11 Debug port (DAP)

Any debug access port that can initiate a reset request from a connected debugger also triggers the DAP reset condition.

The DAP reset condition triggers CMC to assert a warm reset signal to all power domains. A DAP reset sets the appropriate fields in the CMC registers so that software can determine when a DAP reset occurs.

32.4.12 JTAG

The JTAG reset condition triggers whenever a JTAG instruction places the chip in reset.

The JTAG reset condition triggers CMC to assert a warm reset signal to all power domains. A JTAG reset sets the appropriate fields in the CMC registers so that software can determine when a JTAG reset occurs.

32.5 Reset type

This section describes the different resets that can be generated on this chip and their respective reset sequences. There are two types of reset conditions on this chip:

- POR (see POR)
- Warm reset (see Warm reset)

32.5.1 POR

Each power domain generates a POR to reset the entire chip, including debug and mode control logic. The POR for each power domain can assert as a result of the following events:

- Initial power up of the chip
- LVD or HVD
- Power domain wake-up from Power-Down or Deep Power-Down mode (varies by domain)

A POR does not guarantee the contents of on-chip SRAM. Only a Power Down wake-up guarantees the contents of on-chip SRAM, and only for SRAM that you configure to retain state.

32.5.1.1 POR sequence

The following steps are performed as part of the POR (or LVD or HVD) sequence:

1. RESET_b pin is driven low and internal reset signals assert
2. POR and LVD signals negate, and clocks are enabled in their default configuration
3. Internal reset remains asserted for 32 Cortex-M33 core clock cycles
4. Internal reset to Flash and Fuse controller negates and their initialization sequences commence
5. Initialization sequences for Flash and Fuse complete
6. Internal trim registers are loaded and RESET_b pin is tri-stated
7. Reset state machine waits for RESET_b pin input to pull high or drive high externally
8. Internal reset signals negate
9. Core exits reset and fetches the initial program counter and stack pointer from boot ROM

When the chip wakes up from Deep Power-Down or Power-Down mode, the reset sequence differs slightly from the POR sequence. See Deep power-down (DPD) reset sequence and Warm reset for more information.

32.5.1.2 Deep power-down (DPD) reset sequence

The following steps are performed as part of the deep power down wake-up sequence:

1. Internal reset signals assert
2. Wake-up signal negates and clocks are enabled in their default configuration
3. Internal reset remains asserted for 32 clock cycles
4. Internal reset to Flash and Fuse controller negates and their initialization sequences commence
5. Initialization sequences for Flash and Fuse complete
6. Internal trim registers are loaded
7. Internal reset signals negate
8. Core exits reset and fetches the initial program counter and stack pointer from boot ROM

A DPD reset sequence does not assert RESET_B unless you use the RESET_b pin as the wake-up source. A wake-up from deep power down via the RESET_b pin follows the warm reset sequence.

32.5.2 Warm reset

A warm reset reinitializes the chip to a known state after it runs for a period of time.

A warm reset resets most of the logic in each power domain. Warm resets are divided into the following reset sources:

- Fatal
- Nonfatal

You can configure nonfatal reset sources to generate an interrupt instead of the warm reset. If software can clear the nonfatal reset source (including the status flag) within 64 ms, then the warm reset is averted. Nonfatal resets retain the contents of on-chip SRAM.

You cannot configure fatal reset sources to generate an interrupt and they do not guarantee the contents of on-chip SRAM. See the CMC chapter for which reset sources are fatal resets.

The following steps are performed as part of the warm reset sequence:

1. RESET_b pin drives low and internal reset signals assert
2. Clocks are enabled in their default configuration
3. Internal reset remains asserted for 32 clock cycles
4. Internal reset to Flash and Fuse controller negates and their initialization sequences commence
5. Initialization sequences for Flash and Fuse complete
6. Internal trim registers are loaded and RESET_b pin is tri-stated
7. Reset state machine waits for RESET_b pin input to pull high or drive high externally
8. Internal reset signals negate
9. Core exits reset and fetches the initial program counter and stack pointer from boot ROM

Chapter 33
Clocking

33.1 Configuring the main clock and system clock

The clock source for the registers and memories is derived from the main clock. The main clock can be selected from the sources listed in step 1 below.

The main clock, after optionally being divided by the CPU clock divider, is called the system clock. The system clock provides a clock signal to the core, memories, and peripherals (register interfaces and peripheral clocks).

1. Select the main clock in the SCG RCCR register from these options:
   - 12-MHz free-running oscillator (FRO) output (FRO_12M) from internal oscillator.
   - FRO high-speed output (fro_hf) from internal oscillator. By default, its speed is 48 MHz. fro_hf is default main clock.
   - External oscillator.
   - Output of PLL0.
   - Output of PLL1.
   - RTC 32-kHz oscillator.
   - Output of USB PLL (usb_pll_clk).
2. Select the divider value for the system clock AHBCLKDIV register.
3. Enable the clock for the memories and peripherals used in the application.

NOTE

You must disable the peripheral clock (via AHBCLKCTRLx registers) when performing software reset on peripherals (via PRESETCTRLx registers). After asserting and de-asserting reset, you can enable the respective peripheral clock.

33.2 Clock generation

The system control block facilitates clock generation. Many clocking variations are possible. Figure 127 gives an overview of potential clock options. Table 319 describes signals on the clocking diagram. The maximum clock frequency is 150 MHz.

NOTE

The indicated clock multiplexers shown in Figure 127 are synchronized. For the multiplexer to switch gracefully between the two clocks without glitches, the currently selected clock and the clock to be switched to must be running. Other clock multiplexers are not synchronized. If a glitch-free output is needed, the output divider can be stopped and restarted gracefully during switching.

FRO_12M provides a 1-MHz clock (clk_1m) which is independent of watchdog 0. This clock is accurate within a limited range (±3% of each value) of the temperature, voltage, and silicon processing variations made during assembly after trimming. See the data sheet for specifications. To determine the actual watchdog oscillator output, use the frequency measuring block. See Chip-specific Frequency Measurement information.

The device contains two PLLs (PLL0 and PLL1) that can be configured to use a number of clock inputs and produce an output clock. This output clock can then be used to run most on-chip functions. You can monitor the output of the PLL via the CLKOUT pin.

NOTE

The maximum allowed frequency for the main clock and system clock (to CPU0, AHB bus, sync, and others) is 150 MHz.

Table 319. Clocking diagram signal name descriptions

Name	Description
lp_osc	Low power clock source. It is selected as either FRO_16K or XTAL32K in the RTC register.
system_clk	AHB bus clock. Used by the CPU, AHB bus, APB bus, and others. Derived from main_clk.
slow_clk	Slow clock derived from system_clk divided by 4. slow_clk provides the bus clock for FMU, SPC, CMC, TDET, CMP0, CMP1, VBAT, LPTRM0, LPTRM1, RTC, GPIO5, PORT5, and TSI.
clk_in	Internal clock from the external oscillator. clk_in connects to different modules as three functional clock source options. • Ungated, clk_in goes to ADC0, ADC1, DAC0, DAC1, DAC2, CMP0, CMP1, CMP2, ENET, and FlexIO. • Gated by SYSCON.CLOCK_CTRL[1], clk_in goes to FREQME, USBHS, LPTMR0, and LPTMR1 • Gated by SYSCON.CLOCK_CTRL[5], clk_in goes to MICFIL, EMVSIM0, EMVSIM1, CAN0, CAN1, I3C0, I3C1, SAI0, SAI1, SINC Filter (SINC), TSI, USBFS, SCT, uSDHC, and CLKOUT.
CLKOUT	Clock used for debugging purposes or to drive some external logic (see data sheet for limitations on pin output frequency). Many on-chip clocks can be selected to be output on the CLKOUT function. To use CLKOUT, you must connect it to a pin by selecting it in the PORT block.
fro_12m	12-MHz output from FRO_12M.
clk_1m	1-MHz output of FRO_12M.
fro_hf	Clock output from FRO_144M. SCG_FIRCCFG[RANGE] controls the frequency. This clock can only be used for USB devices and is not reliable for the USB host timing requirements for data signaling rate.
fro_hf_div	fro_hf clock, potentially divided by the FRO_HF divider. Used to save power, when fro_hf is faster than needed.
clk_16k	16-kHz clock output from FRO_16K.
main_clk	Main clock used by the CPU and AHB bus, and potentially many others. The main clock and its source selection are shown in Figure 127.
mclk_in	The MCLK input function, when it is connected to a pin by selecting it in the port control block. There are two MCLK signals: mclk_in_0 connects to SAI0, and mclk_in_1 connects to SAI1.
none	Tied-off source. To save power when the output of the related multiplexer is not used, select this source.
pll_clk_div	Output of PLL0 or PLL1, potentially divided. The output may be divided to save power, or because the frequency of the raw PLL output may be too high to be used directly.
pll0_clk	Output of PLL0. PLL0 and its source selection are shown in Figure 127.
pll1_clk	Output of PLL1. PLL1 and its source selection are shown in Figure 127.
wdt_clk	Clock to the watchdog timer.
XTAL32K	Output of the 32-kHz crystal oscillator.

Table 320. Maximum clock speed of modules in run modes

Clock	Maximum speed[1]
clk_in	OD mode: 50 MHz SD mode: 50 MHz

Table continues on the next page...

Table 320. Maximum clock speed of modules in run modes (continued)

Clock	Maximum speed[1]
	MD mode: 50 MHz
fro_hf	OD mode: 144 MHz SD mode: 144 MHz MD mode: 48 MHz
pll0_clk	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
pll1_clk	OD mode: 300 MHz SD mode: 200 MHz MD mode: 100 MHz
usb_pll_clk and ref_pfd	OD mode: 150 MHz SD mode: 100 MHz MD mode: Non-functional
utmi_clk	OD mode: 30 MHz SD mode: 30 MHz MD mode: Non-functional
system_clk	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
CLKOUT	OD mode: 100 MHz SD mode: 64 MHz MD mode: 32 MHz
pll1_clk0	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
pll1_clk1	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
pll_clk_div	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz

Table continues on the next page...

Table 320. Maximum clock speed of modules in run modes (continued)

Clock	Maximum speed[1]
fro_hf_div	OD mode: 144 MHz SD mode: 72 MHz MD mode: 48 MHz
Trace clock	OD mode: 96 MHz SD mode: 96 MHz MD mode: 50 MHz
System tick clock	OD mode: 25 MHz SD mode: 16 MHz MD mode: 8 MHz
FlexSPI serial root clock	OD mode: 300 MHz SD mode: 200 MHz MD mode: 100 MHz
ADC clock	OD mode: 60 MHz SD mode: 48 MHz MD mode: 24 MHz
DAC clock	OD mode: 50 MHz SD mode: 32 MHz MD mode: 16 MHz
CTIMER clock	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
SCTIMER clock	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
OSTIMER clock	OD mode: 1 MHz SD mode: 1 MHz MD mode: 1 MHz
wdt0_clk	OD mode: 1 MHz SD mode: 1 MHz MD mode: 1 MHz
wdt1_clk	OD mode: 24 MHz

Table continues on the next page...

Table 320. Maximum clock speed of modules in run modes (continued)

Clock	Maximum speed[1]
	SD mode: 24 MHz MD mode: 8 MHz
USBFS clock	OD mode: 48 MHz SD mode: 48 MHz MD mode: 48 MHz
EMVSIM clock	OD mode: 50 MHz SD mode: 32 MHz MD mode: 16 MHz
CMP clock	OD mode: 25 MHz SD mode: 16 MHz MD mode: 8 MHz
LP_FLEXCOMM0/1/2 clock	OD mode: 75 MHz SD mode: 50 MHz MD mode: 25 MHz
LP_FLEXCOMM3/4/5 clock	OD mode: 100 MHz SD mode: 100 MHz MD mode: 50 MHz
LP_FLEXCOMM6/7/8/9 clock	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
CAN clock	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
FlexIO clock	OD mode: 150 MHz SD mode: 100 MHz MD mode: 50 MHz
uSDHC clock	OD mode: 104 MHz SD mode: 100 MHz MD mode: 50 MHz
SINC Filter clock	OD mode: 150 MHz SD mode: 100 MHz

Table continues on the next page...

Table 320. Maximum clock speed of modules in run modes (continued)

Clock	Maximum speed[1]
	MD mode: 50 MHz
I3C FCLK	OD mode: 25 MHz SD mode: 25 MHz MD mode: 25 MHz
CLK_SLOW_TC	OD mode: 25 MHz SD mode: 25 MHz MD mode: 25 MHz
SAI clock	OD mode: 50 MHz SD mode: 32 MHz MD mode: 16 MHz
ENET RMII clock	OD mode: 50 MHz SD mode: 50 MHz MD mode: Not functional
ENET PTP clock	OD mode: 150 MHz SD mode: 100 MHz MD mode: Not functional
MICFIL clock	OD mode: 50 MHz SD mode: 32 MHz MD mode: 16 MHz
TSI clock	OD mode: 25 MHz SD mode: 25 MHz MD mode: 25 MHz

1. OD = Over Drive, SD = Standard Drive, MD = Mid Drive

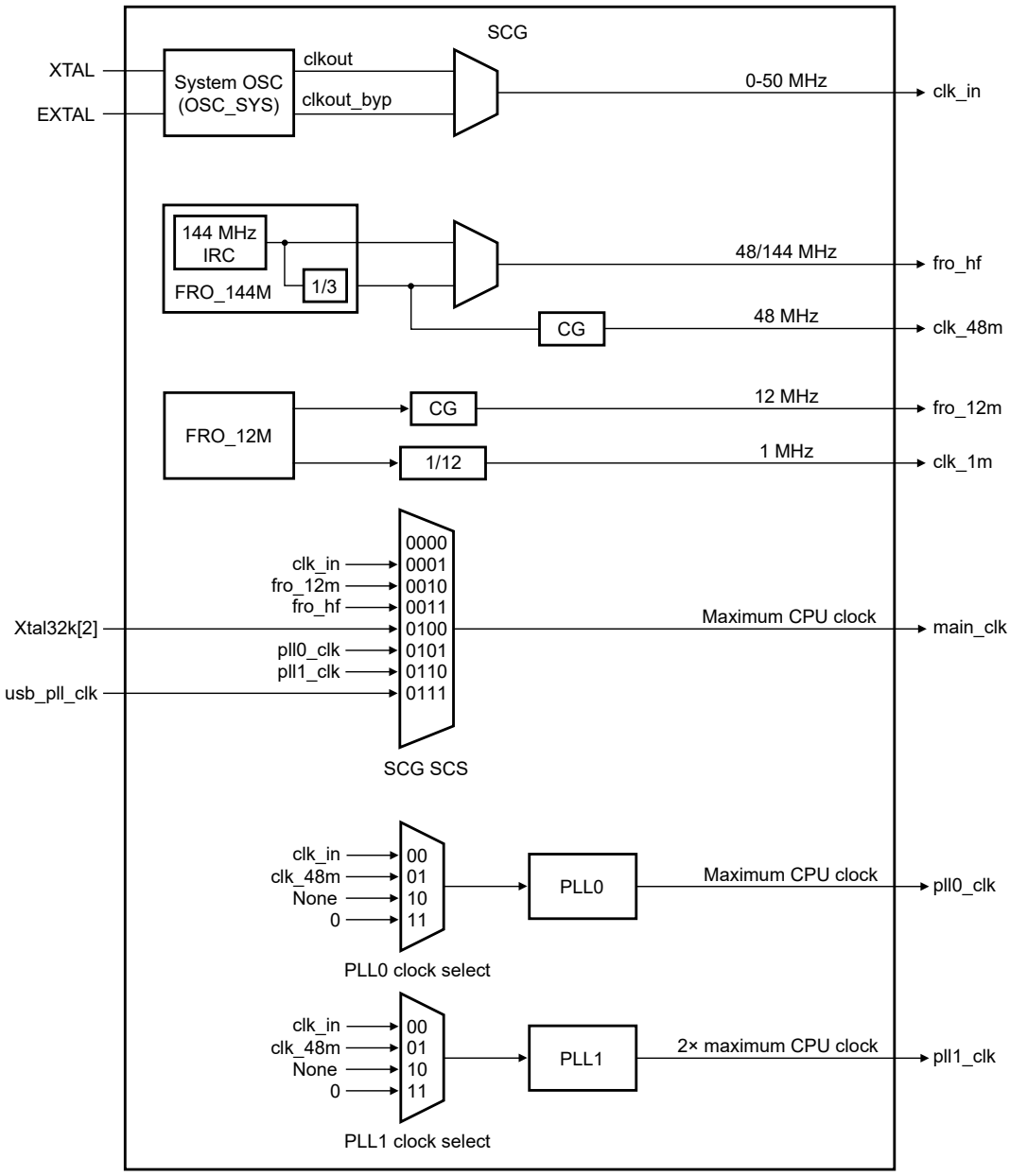


Figure 127. SCG main clock

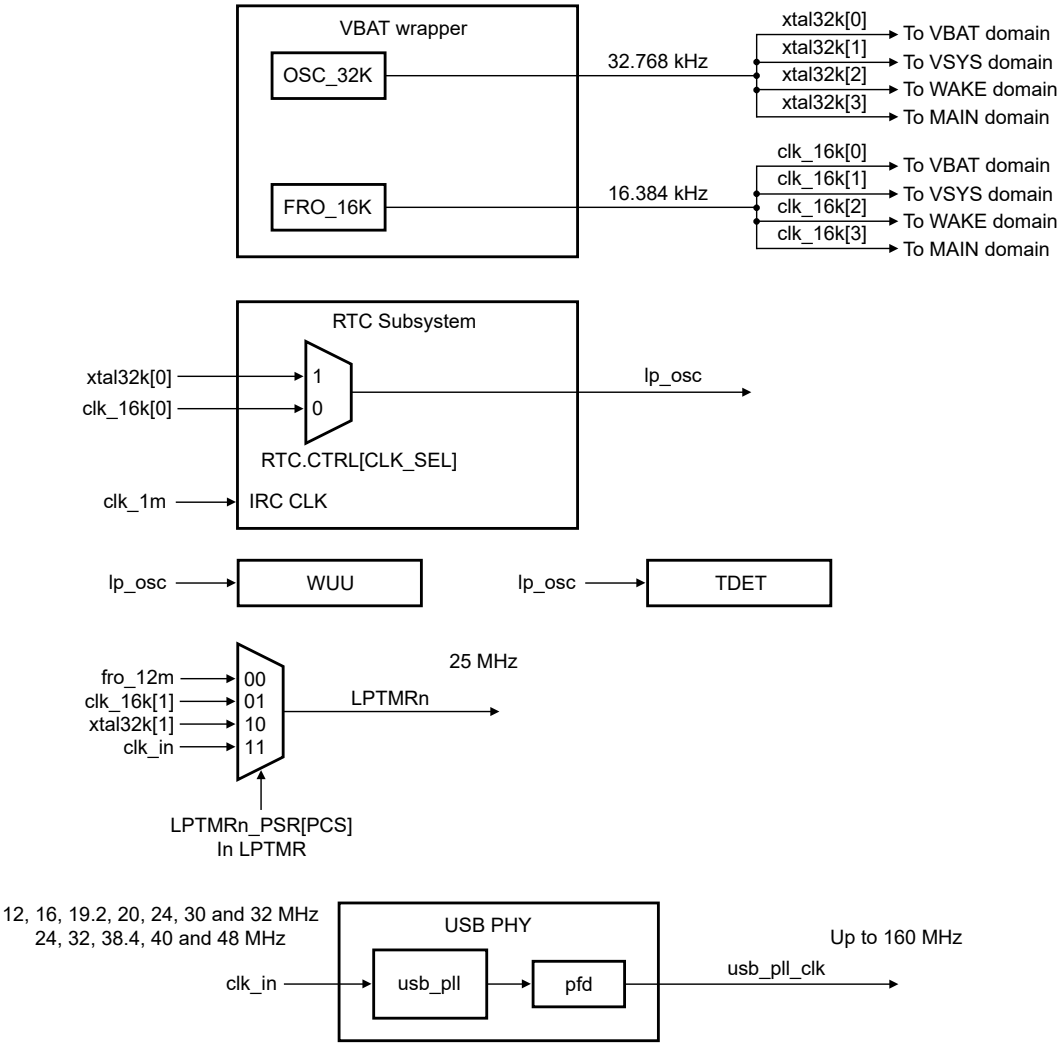


Figure 128. Clock generation

33.3 Module clocking

These sections show the clock generation diagrams for the modules of the chip.

33.3.1 CPU clock divider

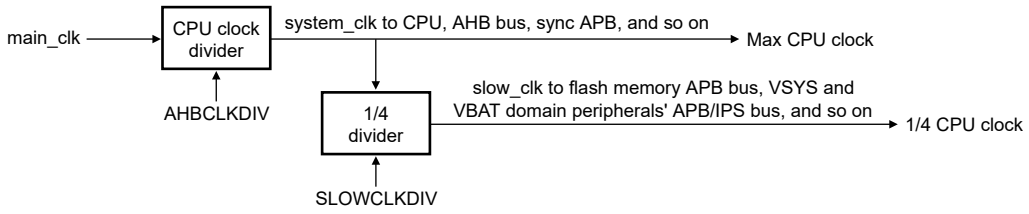


Figure 129. Main CPU clocking diagram

33.3.2 CLKOUT clock divider

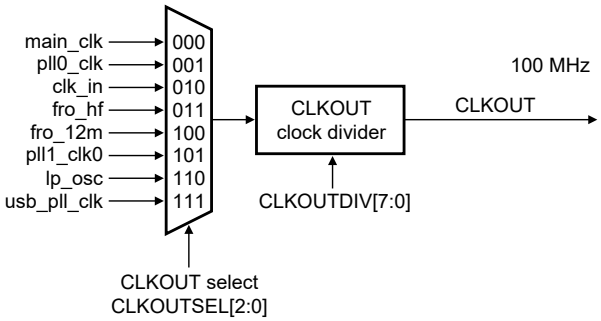


Figure 130. CLKOUT diagram

33.3.3 PLL1 clock divider

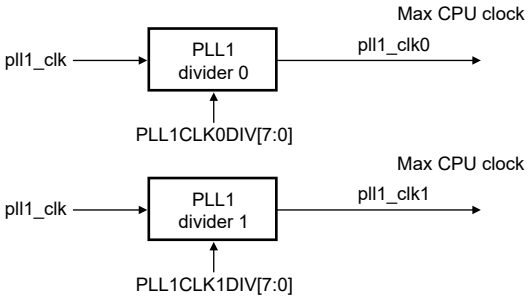


Figure 131. PLL1 divider diagram

33.3.4 PLL clock divider

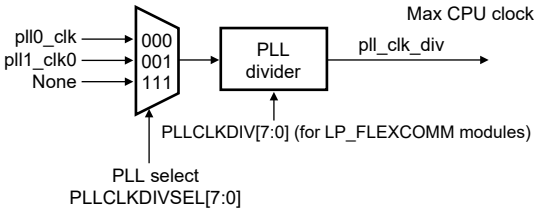


Figure 132. PLL clock divider diagram

33.3.5 FRO_HF clock divider

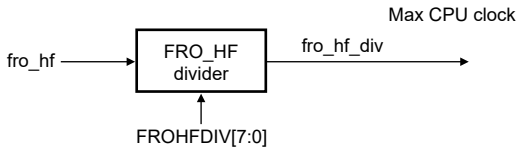


Figure 133. FRO_HF clock diagram

33.3.6 Trace clock divider clocking

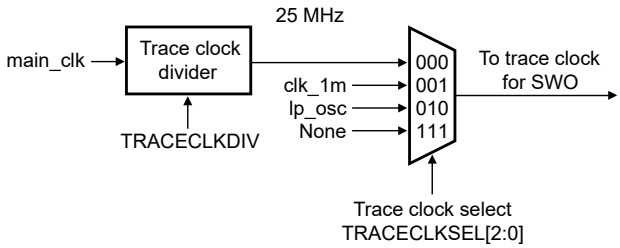


Figure 134. Trace clock divider diagram

33.3.7 Systick clocking

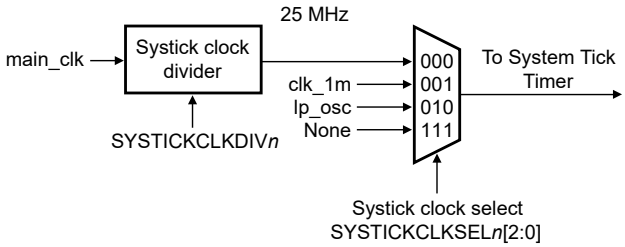


Figure 135. Systick clock select diagram

33.3.8 FlexSPI clocking

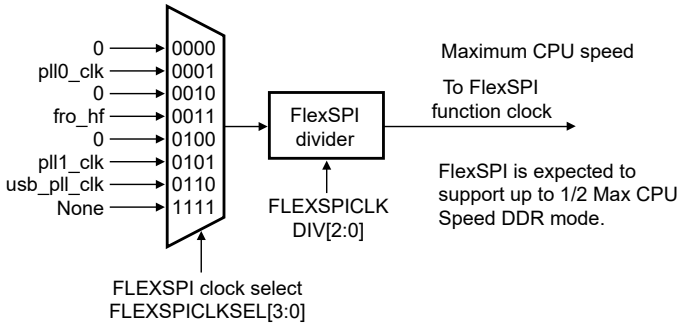


Figure 136. FlexSPI clocking diagram

33.3.9 ADC clocking

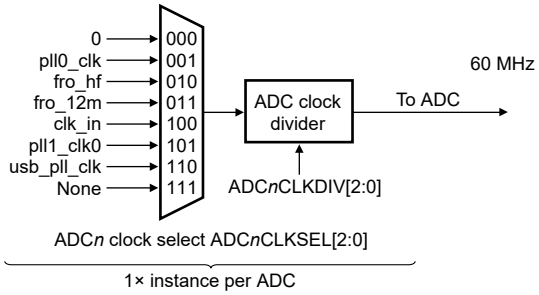


Figure 137. ADC clocking diagram

33.3.10 DAC clocking

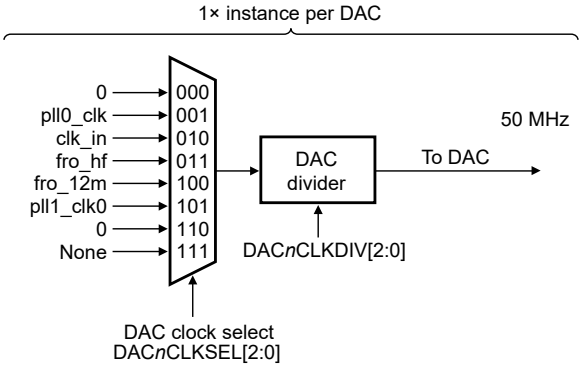


Figure 138. DAC clocking diagram

33.3.11 CTIMER clocking

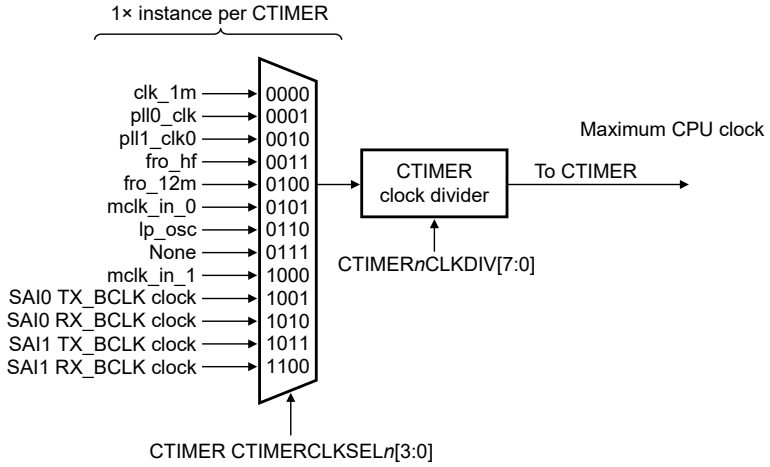


Figure 139. CTIMER clocking diagram

33.3.12 SCTIMER clocking

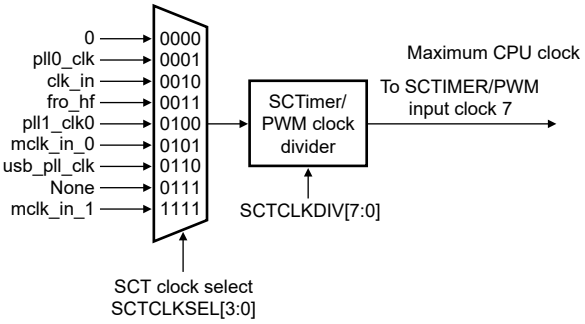


Figure 140. SCTIMER clock diagram

33.3.13 UTICK clocking

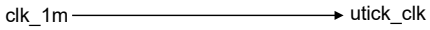


Figure 141. UTICK clocking diagram

33.3.14 OSTIMER clocking

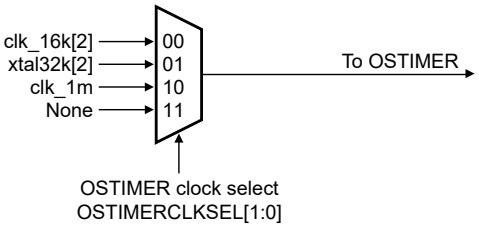


Figure 142. OSTIMER clocking diagram

33.3.15 QDC clocking

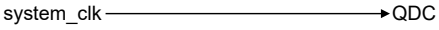


Figure 143. QDC clocking diagram

33.3.16 PWM clocking

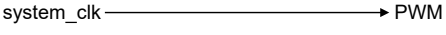


Figure 144. PWM clocking diagram

33.3.17 NPU clocking

Figure 145. NPU clocking diagram

33.3.18 EWM clocking

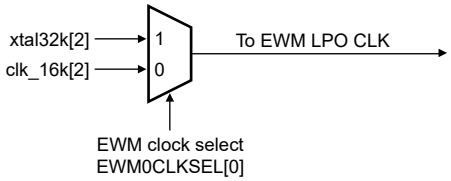


Figure 146. EWM clock diagram

33.3.19 WWDT clocking

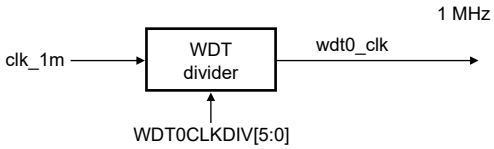


Figure 147. WWDT0 clock diagram

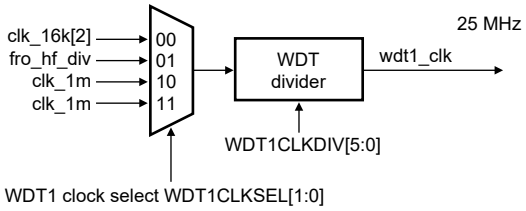


Figure 148. WWDT1 clock diagram

33.3.20 USBFS clocking

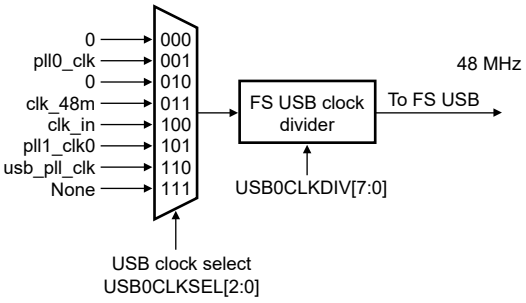


Figure 149. USBFS clocking diagram

33.3.21 EMVSIM clocking

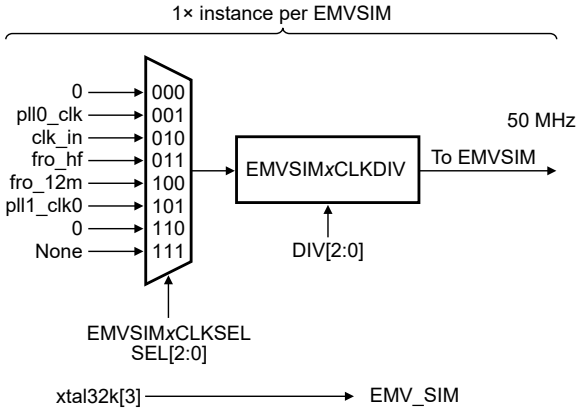
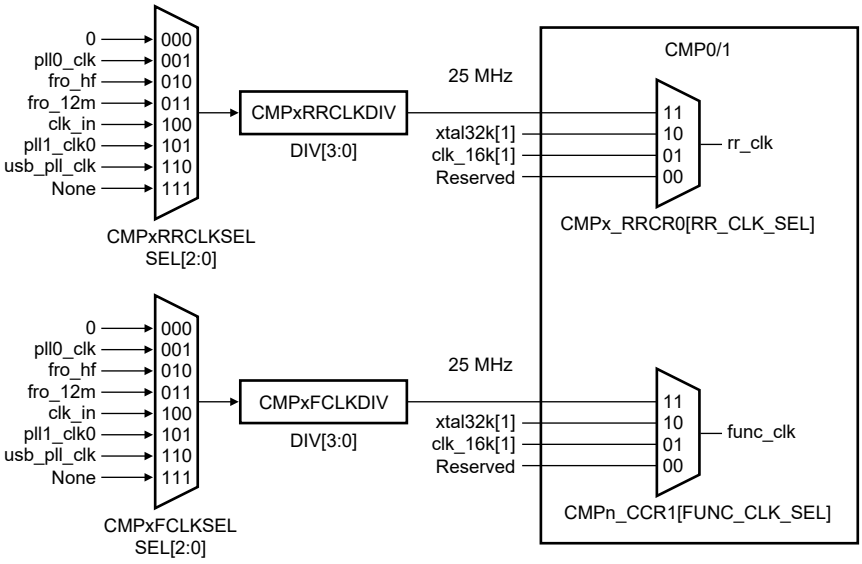


Figure 150. EMVSIM clocking diagram

33.3.22 CMP clocking



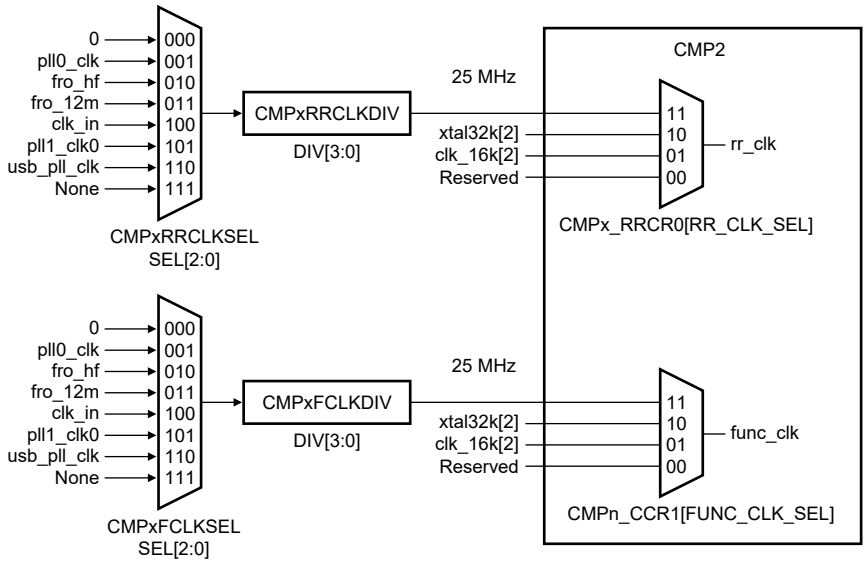


Figure 151. CMP clocking diagram

33.3.23 LP_FLEXCOMM clocking

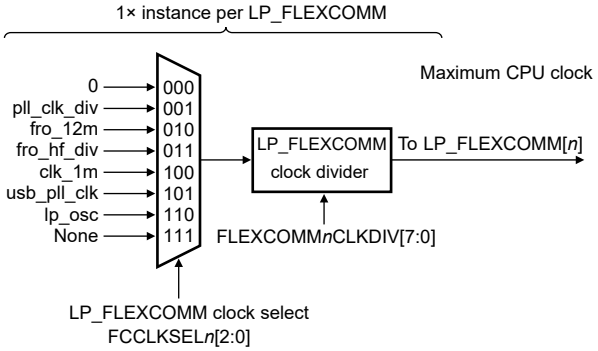


Figure 152. LP_FLEXCOMM clocking diagram

33.3.24 FlexCAN clocking

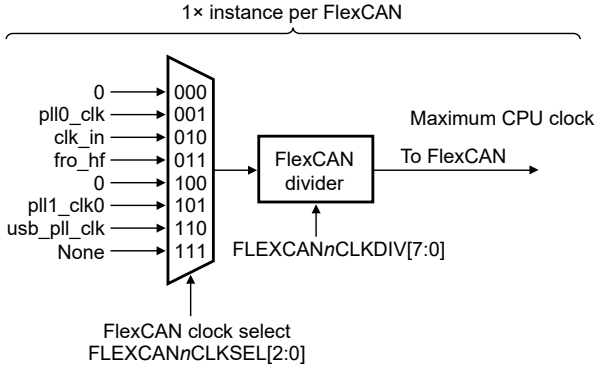


Figure 153. FlexCAN clocking diagram

33.3.25 FlexIO clocking

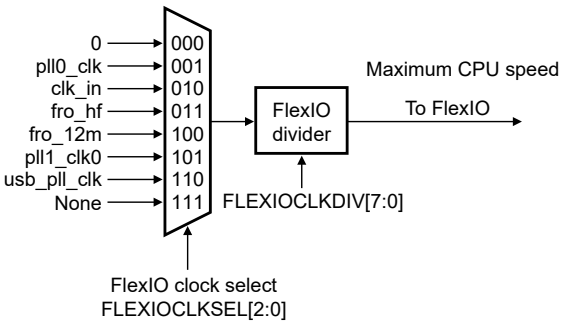


Figure 154. FlexIO clocking diagram

33.3.26 uSDHC clocking

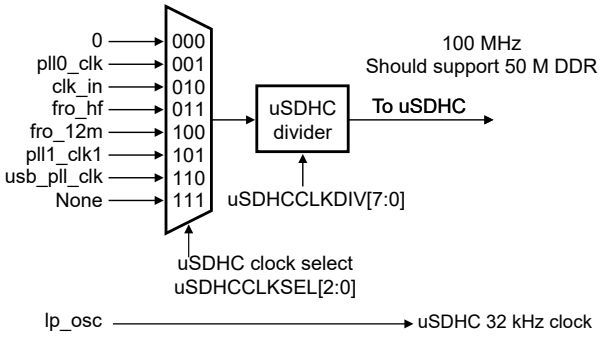


Figure 155. uSDHC clocking diagram

33.3.27 SINC Filter clocking

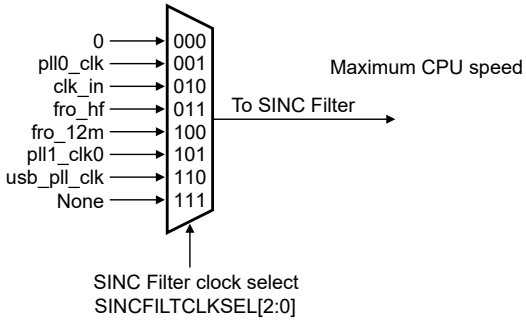


Figure 156. SINC Filter clocking diagram

33.3.28 I3C clocking

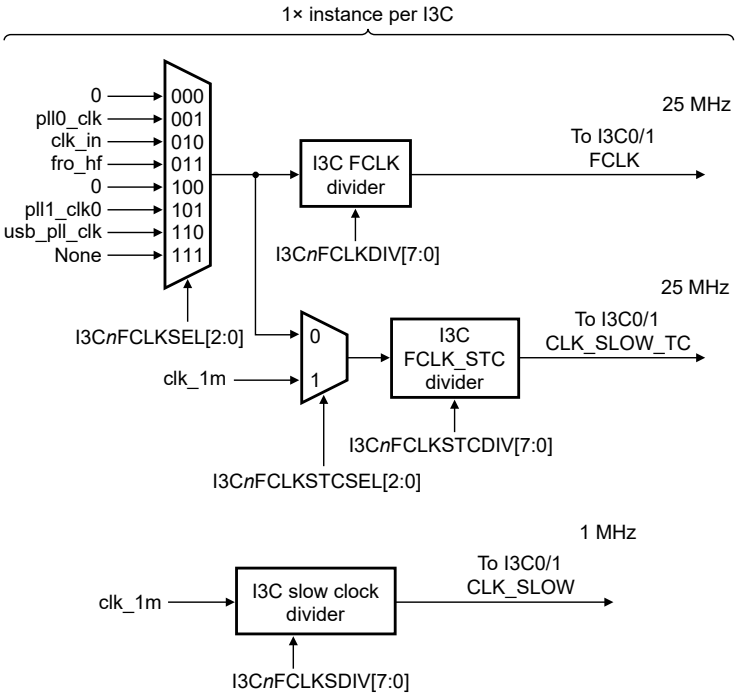


Figure 157. I3C clocking diagram

33.3.29 SAI clocking

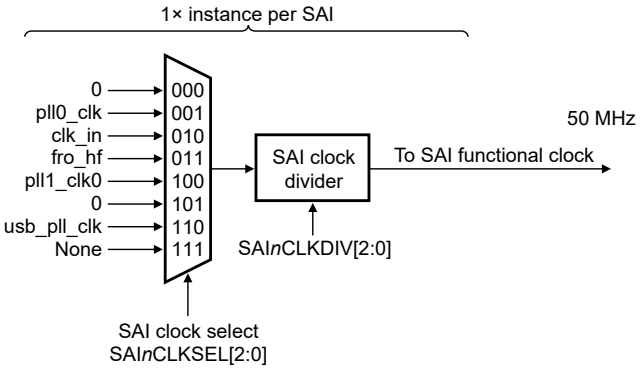


Figure 158. SAI clock diagram

33.3.30 ENET clocking

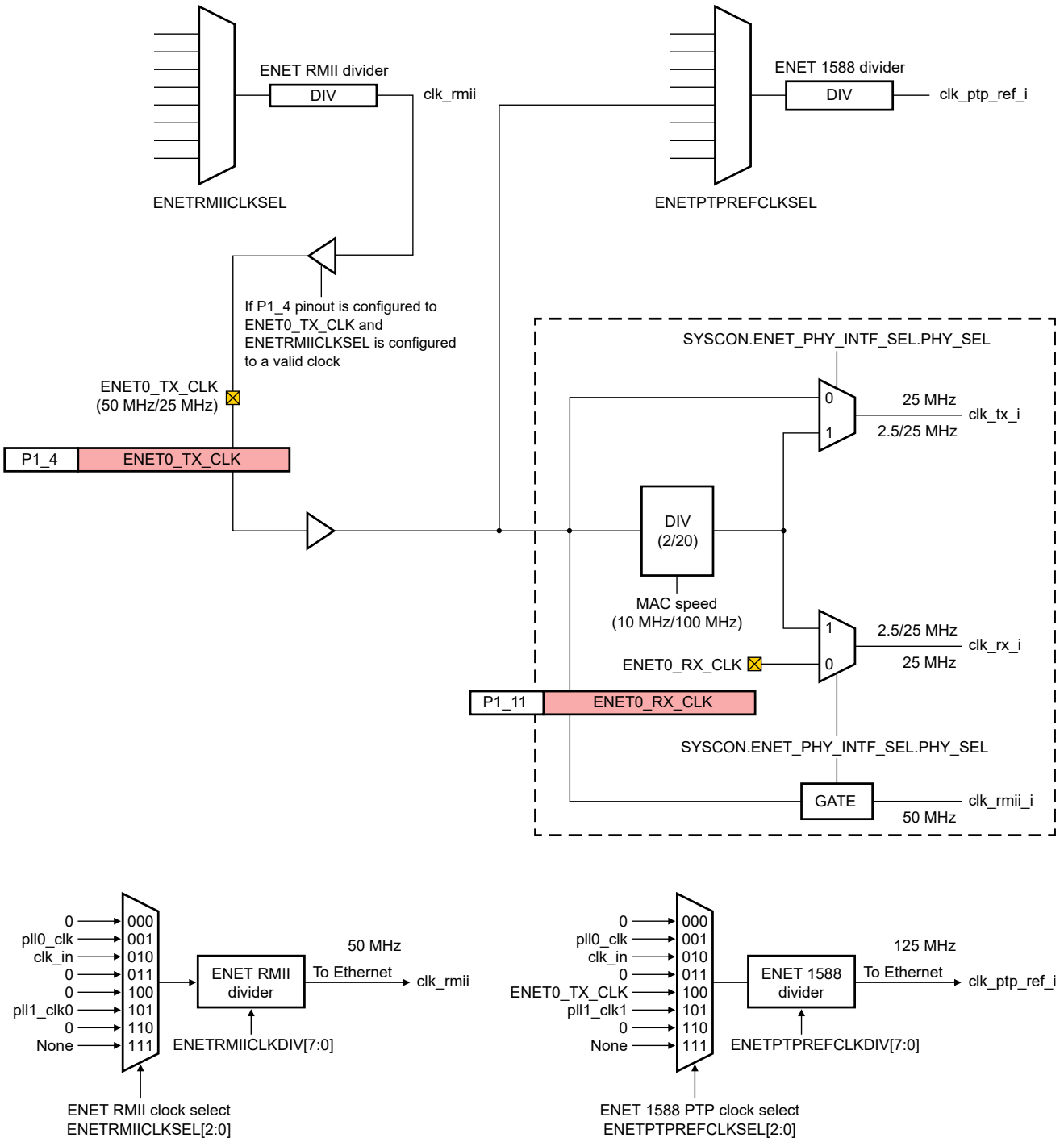


Figure 159. ENET clocking diagram

33.3.31 MICFIL clock divider clocking

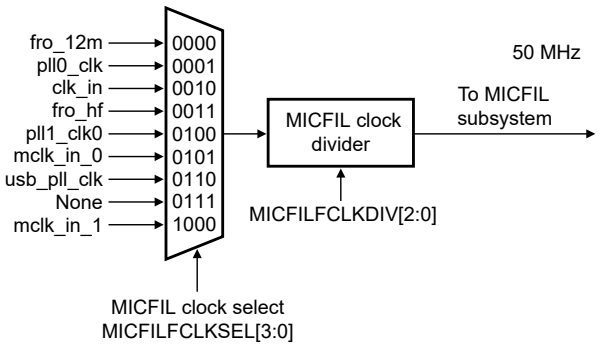


Figure 160. MICFIL clock divider diagram

33.3.32 TSI clocking

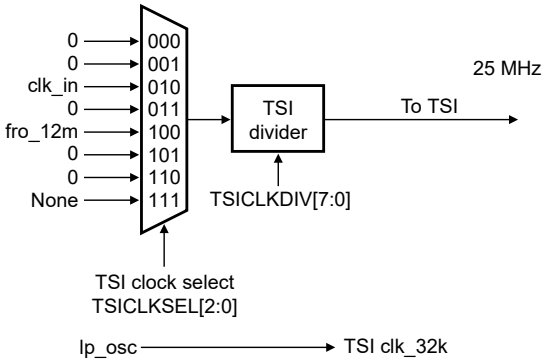


Figure 161. TSI clocking diagram

33.3.33 GDET clocking

Figure 162. GDET clocking diagram

33.4 Set up PLL0

PLL0 creates a stable output clock at a higher frequency than the input clock. If you need a main clock with a frequency higher than the 12-MHz FRO clock and the 144-MHz FRO clock (fro_hf) is not appropriate, use the PLL to boost the input frequency.

33.5 Set up PLL1

PLL1 creates a stable output clock at a higher frequency than the input clock. If you need a main clock with a frequency higher than the 12-MHz FRO clock and the 144-MHz FRO clock (fro_hf) is not appropriate, use the PLL to boost the input frequency.

Chapter 34
System Clock Generator (SCG)

34.1 Chip-specific SCG information

Table 321. Reference links to related information

Topic	Related module	Reference
Full description	SCG	SCG
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

34.1.1 Module instances

This device contains one instance of the SCG module, SCG0.

34.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

34.1.3 Crystal oscillator clock sources

In this device, the ROSC is sourced from Crystal Oscillator 32.768 kHz clock in VBAT block, and the SOSC is sourced from System Crystal Oscillator Clock in the SCG block.

34.1.4 SIRCCSR[SIRCSTEN] power domain

SIRCCSR[SIRCSTEN] is enabled when the WAKE domain is in Deep Sleep mode. When the WAKE domain is in Deep Sleep mode, PMCTRLWAKE[LPMODE]=0x1.

34.1.5 Clock name decoder ring

The table below shows translation of clock name sources used in the SCG chapter to the names used throughout the document.

Table 322. Clock name decoder ring

Clock source specified in chapter	Decoded clock name
APLL	PLL0_CLK
SPLL	PLL1_CLK
SOSC	CLK_IN
SIRC	FRO_12M
FIRC	FRO_HF
ROSC	XTAL32K
UPLL	USB_PLL_CLK

34.2 Overview

The System Clock Generator (SCG) provides the main clock and additional peripheral clocks for the MCU. SCG takes clock inputs from a variety of sources and generates the clocks that the MCU requires.

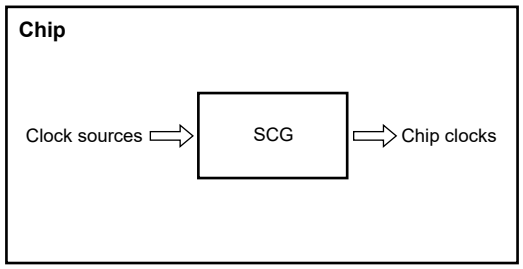


Figure 163. System Clock Generator (SCG) in MCU

34.2.1 Block diagram

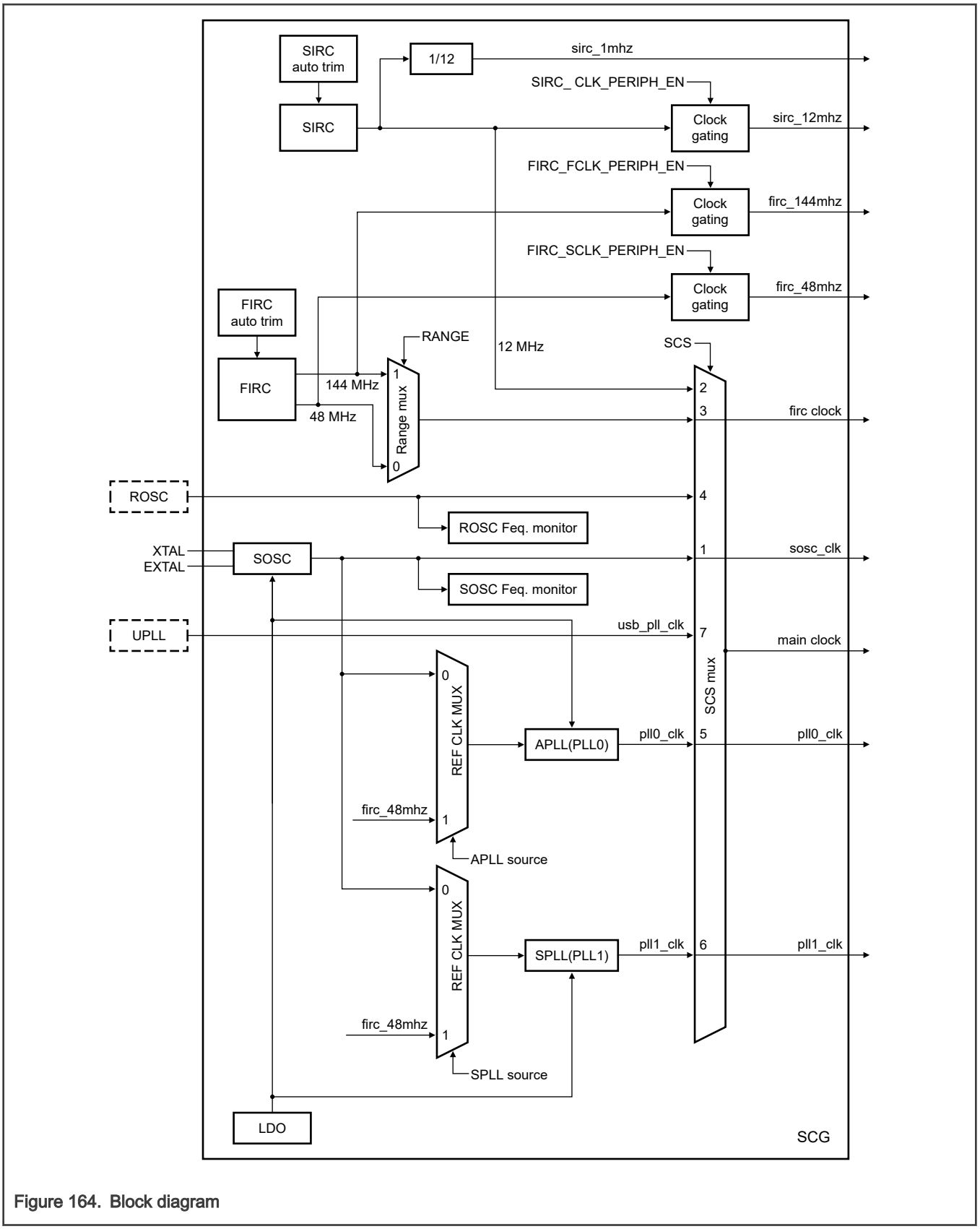


Figure 164. Block diagram

34.2.2 Features

Key features of SCG are:

- Auxiliary phase-locked loop (APLL) clock and system phase-locked loop (SPLL) clock. See PLL features and benefits and Functional features for details.
  - Can be selected as the clock source for system clocks
- Two internal reference clock (IRC) generators with trim capability: FIRC and SIRC
  - FIRC can output 144 MHz or 48 MHz clock by range select and can be used as a source for the APLL and SPLL.
  - SIRC can output 12 MHz clock.
  - Either of the IRC generators can be selected as the clock source for the system clocks.
- System crystal oscillator clock (SOSC)
  - Can be used as a source for the APLL and SPLL.
  - Can be selected as the clock source for system clocks.
- Real-time oscillator clock (ROSC)
  - Can be selected as the clock source for the device. To ascertain which signal the ROSC input maps to, see the chip-specific information.
- Clock monitor with reset and interrupt request capability for SOSC, ROSC, APLL, SPLL, and UPLL clocks
- Low dropout regulator (LDO) used as the supply for the PLL and SOSC
  - Configurable output voltage (1 V to 1.25 V)
  - Wide supply range (1.71 V up to 3.63 V)
  - Wide load current range (up to 8 mA)
  - Low noise, high Power Supply Rejection Ratio (PSRR)
  - Bypass mode

34.3 Functional description

34.3.1 SCG clock mode transitions

The system boots up from the FIRC source. The system clock can be switched between the following clock sources: FIRC, SIRC, SOSC, ROSC, APLL, SPLL, and UPLL.

SCG restricts programming into invalid clock modes, and ignores writes to the System Clock Source (SCS) fields. When a transition between Run modes or a transition from Run mode into Wait mode occurs, SCG switches to the corresponding clock mode as defined in the Run Clock Control Register (RCCR). After the switch is complete, the system switches to the selected Run or Wait mode.

The following table shows the SCG modes of operation.

Table 323. SCG modes of operation

Mode	Description
SOSC	SCG enters SOSC mode when the following occurs: • The device is in Run mode and 0001 is written to RCCR[SCS] • SOSCCSR[SOSCEN] = 1b

Table continues on the next page...

Table 323. SCG modes of operation (continued)

Mode	Description
	• SOSCCSR[SOSCVLD] = 1b In SOSC mode, the system clocks are derived from the external SOSC.
SIRC	SCG enters SIRC mode when the following occurs: • The device is in Run mode and 0010 is written to RCCR[SCS] • SIRCCSR[SIRCVLD] = 1b In SIRC mode, the system clocks are derived from SIRC.
FIRC	FIRC mode is the default clock mode of operation. SCG enters FIRC mode when the following occurs: • The device is in Run mode and 0011 is written to RCCR[SCS] • FIRCCSR[FIRCEN] = 1b • FIRCCSR[FIRCVLD] = 1b In FIRC mode, the system clocks are derived from FIRC. Two frequency range settings are available for FIRC clock, as described in FIRCCFG[RANGE]. Changes to FIRC range settings are ignored when the FIRC clock is enabled.
ROSC	SCG enters ROSC mode when the following occurs: • The device is in Run mode and 0100 is written to RCCR[SCS] • ROSCCSR[ROSCVLD] = 1b • LDOCSR[LDOEN] = 1b and LDOCSR[VOUT_OK] = 1b In ROSC mode, the system clocks are derived from the external ROSC.
APLL	SCG enters APLL mode when the following occurs: • The device is in Run mode and 0101 is written to RCCR[SCS] • APLLCSR[APLLCLKEN] = 1b • APLLCSR[APLLPWREN] = 1b • APLLCSR[APLL_LOCK] = 1b • LDOCSR[LDOEN] = 1b and LDOCSR[VOUT_OK] = 1b In APLL mode, the system clocks are derived from the output of the PLL which is controlled by either the FIRC or SOSC. The selected PLL clock frequency locks to a multiplication factor (as specified by its corresponding MDIV) times the selected PLL reference frequency. The programmable reference divider of the PLL must be configured to produce a valid PLL reference clock. This divide value is defined by the NDIV bits.
SPLL	SCG enters SPLL mode when the following occurs: • The device is in Run mode and 0110 is written to RCCR[SCS] • SPLLCSR[SPLLCLKEN] = 1b • SPLLCSR[SPLLPWREN] = 1b • SPLLCSR[SPLL_LOCK] = 1b

Table continues on the next page...

Table 323. SCG modes of operation (continued)

Mode	Description
	• LDOCSR[LDOEN] = 1b and LDOCSR[VOUT_OK] = 1b In SPLL mode, the system clocks are derived from the output of PLL which is controlled by either the FIRC or SOSC. The selected PLL clock frequency locks to a multiplication factor (as specified by its corresponding MDIV) times the selected PLL reference frequency. The programmable reference divider of the PLL must be configured to produce a valid PLL reference clock. This divide value is defined by the NDIV bits.
UPLL	SCG enters UPLL mode when the following occurs: • The device is in Run mode and 0111 is written to RCCR[SCS] • UPLLCSR[UPLLVLD] = 1b In UPLL mode, the system clocks are derived from the output of UPLL.
Stop	SCG enters Stop mode whenever the CORE domain enters a low power, state retention, or power down state. Power modes are chip-specific. For power mode assignments, see Power Management chapter that describes module operation in low-power modes. All SCG clock signals are static when entering Stop mode, including SCG system clocks (except clocks explicitly configured to continue running and are able to continue running in the target power mode). There are some exceptions. The following clocks can continue to run and stay enabled when all the respective conditions listed below are true. • SIRC is available in Deep Sleep mode when all the following conditions are true: — SIRCCSR[SIRCSTEN] = 1b • FIRC is available in Deep Sleep mode when all the following conditions are true: — FIRCCSR[FIRCEN] = 1b — FIRCCSR[FIRCSTEN] = 1b • SOSC is available in Deep Sleep mode when all the following conditions are true: — SOSCCSR[SOSCEN] = 1b — SOSCCSR[SOSCSTEN] = 1b — LDOCSR[LDOEN] = 1b and LDOCSR[VOUT_OK] = 1b • APLL is available in Deep Sleep mode when all the following conditions are true: — APLLCSR[APLLCLKEN] = 1b — APLLCSR[APLLPWREN] = 1b — APLLCSR[APLLSTEN] = 1b — LDOCSR[LDOEN] = 1b and LDOCSR[VOUT_OK] = 1b • SPLL is available in Deep Sleep mode when all the following conditions are true: — SPLLCSR[SPLLCLKEN] = 1b — SPLLCSR[SPLLPWREN] = 1b — SPLLCSR[SPLLSTEN] = 1b — LDOCSR[LDOEN] = 1b and LDOCSR[VOUT_OK] = 1b

34.3.2 SCG SOSC operation

The following table lists the SOSC configurations and SOSC output clock.

Table 324. SOSC mode operation

SOSCEN	EREFS	SOSC clock
0	0	Off
1	0	EXTAL Pad Clock
1	1	Crystal Oscillator Clock

34.3.3 SCG LOC operation

SCG supports the loss of clock (LOC) monitoring for the external reference clocks, including SOSC and ROSC. To enable the LOC monitor, the CMRE field for each of these clocks must be programmed to 1. To monitor these external reference clocks, the SCG uses an internal 31.25KHz reference clock derived by dividing the SIRC internal clock. You must program the SIRC to be enabled prior to enabling the LOC for either of these external reference clocks.

NOTE

F_{int} = SIRC divided down to 31.25 KHz

When enabled, the LOC monitors generate a system reset or system interrupt request depending on the CMRE field settings (see the CMRE description for each of the clocks for further information). The frequency conditions for the external reference clocks that trigger a LOC event are listed below:

- ROSC loss of external clock minimum frequency
  - When ROSC clock frequency is above (3/5) F_{int}, the chip is never reset and will not generate an interrupt.
  - When ROSC clock frequency is between (2/5) F_{int} and (3/5) F_{int}, the chip might reset or generate an interrupt, depending on the phase dependency between the ROSC clock and the reference clock.
  - When ROSC clock frequency is below (2/5) F_{int}, the chip always resets or generates an interrupt.
- SOSC loss of external clock minimum frequency
  - When SOSC clock frequency is above (16/5) F_{int}, the chip is never reset and will not generate an interrupt.
  - When SOSC clock frequency is between (15/5) F_{int} and (16/5) F_{int}, the chip might reset or generate an interrupt, depending on the phase dependency between the SOSC clock and the reference clock.
  - When SOSC clock frequency is below (15/5) F_{int}, the chip always resets or generates an interrupt.

34.3.4 APLL and SPLL operation

APLL and SPLL have similar functions and registers.

PLL is typically used to create a frequency that is higher than other on-chip clock sources and to operate both the CPU and other on-chip functions. You can also use it to obtain a specific clock that is otherwise not available. For example, a source clock with a frequency of any integer MHz (such as the 12 MHz FIRC) can be divided down to 1 MHz, then multiplied up to any other integer MHz (such as 13, 14, and 15).

PLL can be set up by calling an API supplied by NXP Semiconductors. See APLL and SPLL registers in SCG register descriptions for more information.

NOTE

APLL and SPLL have similar registers. Any references to the register or field names of these registers, without prefix "A" or "S", apply for both registers. For example, PLL_LOCK refers to both APLLCSR[APLL_LOCK] and SPLLCSR[SPLL_LOCK].

34.3.4.1 General description

PLL is a multipurpose phase-locked loop (PLL). It generates a CMOS-compatible clock and can be used in many applications, including as a frequency synthesizer and clock for digital circuits, and analog-to-digital and digital-to-analog converters.

PLL includes several normal modes, spread spectrum mode, fractional mode, and a power-down mode. These are summarized in Table 325 and detailed in the following sections.

Table 325. PLL operating mode summary

Mode	PLLCSR[PLLPWREN]	PLLSSCG1[SS_PD]	PLLCTRL[LIMUPOFF]	PLLSSCG1[SEL_SS_MDIV]	PLLSSCG1[MR[2:0]]	M comes from
Normal	1	1	0	0	0	MDIV[15:0]
Fractional	1	0	1	1	0	SS_MDIV[32:0]
Spread spectrum	1	0	1	x	>0	MDIV[15:0] or SS_MDIV[32:0]
Power-down	0	1	x	x	x	x

34.3.4.2 Abbreviations

Table 326. Abbreviations and notations

Acronym	Description
PLL	Phase-locked loop
PFD	Phase frequency detector
CCO	Current control oscillator
PSR	Power supply rejection
EMI	Electromagnetic interferences
SSCG	Spread spectrum clock generation
clkin	Input clock to the PLL
F_{in}	Frequency of clkin
clkout	Output clock of the PLL
F_{out}	Frequency of clkout, $F_{out} = F_{cco} / (2 * P)$ • $F_{out} = F_{cco} / (2 * P)$ (The divide-by-2 divider in the postdivider is not bypassed.) • $F_{out} = F_{cco} / P$ (Bypass divide-by-2 divider in the postdivider.)
clkref	PLL reference clock. The input clock to the PFD.
F_{ref}	Frequency of clkref, $F_{ref} = F_{in} / N$

Table continues on the next page...

Table 326. Abbreviations and notations (continued)

Acronym	Description
F_{cco}	Frequency of output clock of the CCO, $F_{cco} = M * F_{ref}$
N	Predivider value
M	Feedback divider value
P	Postdivider value
t_{pon}	PLL start-up time

34.3.4.3 PLL block diagram

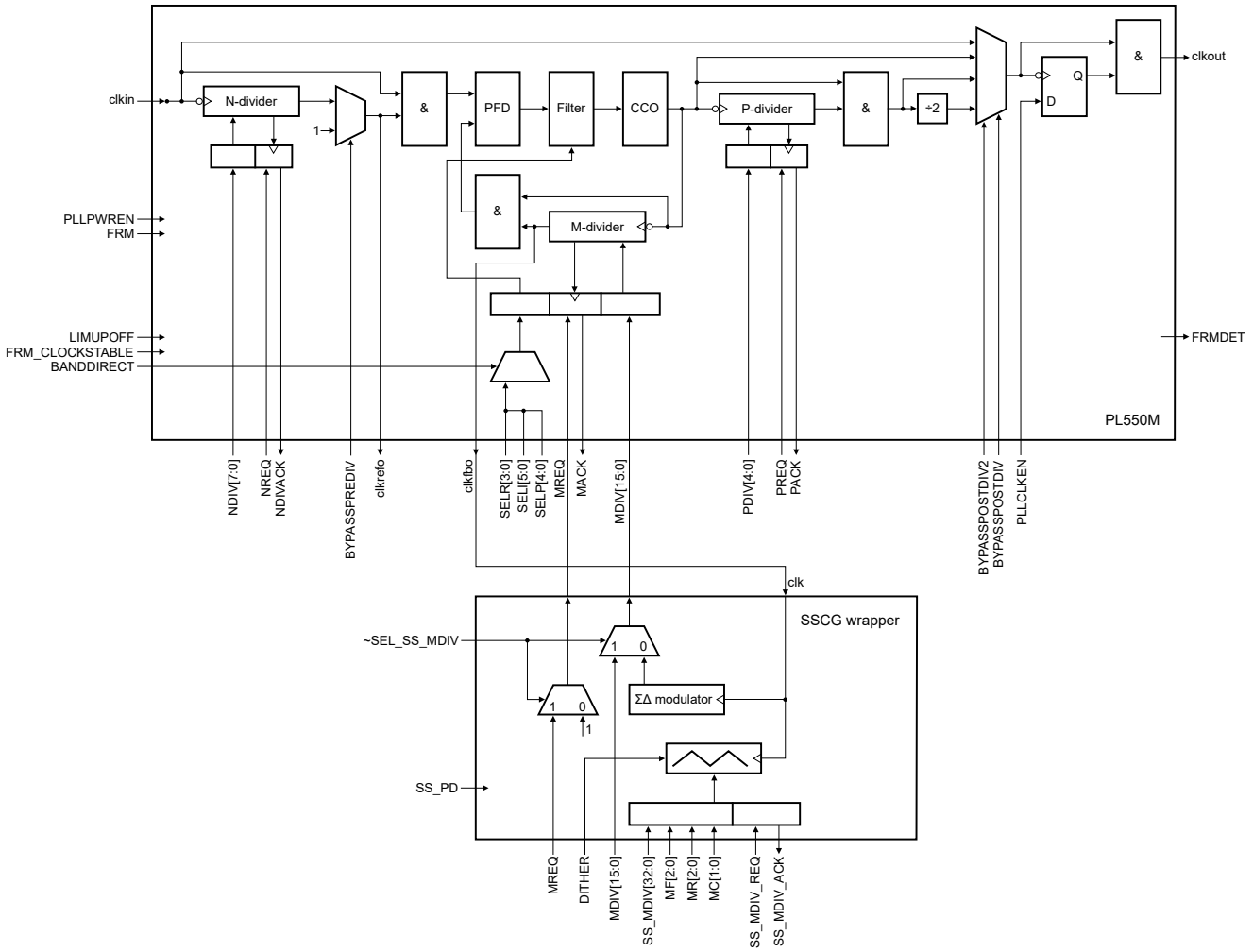


Figure 165. PLL block diagram

34.3.4.4 PLL normal mode

34.3.4.4.1 PLL features and benefits

- Integrated PLL with no external components for clock generation

- Large input range at the phase detector (F_{ref})
- Programmable:
  - Predivider N (N, 1 to 2^8-1)
  - Feedback-divider M (M, 1 to 2^{16}-1)
  - Postdivider P * 2 (P, 1 to 2^5-1)
- Programmable bandwidth (integrating action, proportional action, high frequency pole)
- On-the-fly adjustment of the clock dividers with handshake control
- Positive edge clocking
- Frequency limiter to avoid PLL hang up
- Power down mode
- Support for bypassing the predivider and the postdivider
- Support for bypassing the divide-by-2 in the postdivider
- Support for disabling the output clock
- Free running mode

34.3.4.4.2 PLL functional description

34.3.4.4.2.1 Basic functions

PLL is designed to generate a CMOS compatible clock. PLL can also be used to filter clocks or to make spread spectrum and fractional clocks. The following sections detail these features.

34.3.4.4.2.2 Main blocks

The PLL block diagram is shown in Figure 165. The clock input must be fed to clkin; clkout is the PLL clock output. The analog part of the PLL consists of a phase frequency detector (PFD), a filter, and a current controlled oscillator (CCO). The PFD has two inputs: a reference input from the (divided) external clock and one input from the divided CCO output clock. The PFD compares the phase and frequency of these input signals and generates a control signal if they do not match. This control signal is fed to a filter which drives the CCO.

PLL contains three programmable dividers: predivider (N), feedback-divider (M), and postdivider (P). Every divider contains a bus (XDIV[n:0] in which X is N, M, or P) to load a divider ratio. The dividers also possess the handshake signals, XREQ, and XACK, to select a new divider ratio. See Selecting divider ratios for more information on selecting the divider ratios.

PLL is stable after startup time t_{pon}. There is a hardware timer in the reference clock domain in SCG to count this time. You must configure the LOCK_TIME register to (500 µs/T_{ref} + 300). When the number of reference clocks reaches LOCK_TIME after power on or reconfiguration, the flag PLL_LOCK is set. If the interrupt enable bit PLL_LOCK_IE is set, a PLL_LOCK interrupt is generated.

NOTE

The PLL clock is gated off when PLL_LOCK is low, regardless of the state of the PLLCLKEN field. At start, set PLLPWREN and PLLCLKEN to 1 to enable the PLL. After t_{pon}, the PLL_LOCK is valid and the PLL clock is automatically ungated.

To avoid frequency hang-up, the PLL contains a frequency limiter. This feature is built in to prevent the CCO from running too fast. This can occur if, for example, a wrong feedback-divider (M) ratio is applied to the PLL.

There are additional dividers in the clocking system to bring the PLL output frequency down to what is needed for the CPU and peripherals.

34.3.4.4.3 PLL application information

For more information on the selectable bandwidth, input clocks, and divider ratios of the PLL, see the following sections:

- Selecting bandwidth
- Input clock
- Selecting divider ratios

PLL has eight modes:

- Mode 1: Normal operating mode
- Mode 2: Reserved
- Mode 3: Power down mode (PLLPWREN)
- Mode 4: Reserved
- Mode 5: Free running mode (FRM)
- Mode 6: Reserved
- Mode 7: Reserved
- Mode 8: Enable mode (PLLCLKEN).

34.3.4.4.3.1 Selecting bandwidth

For some applications, such as filtering of the input signal, it can be advantageous to change the bandwidth of the PLL.

In normal applications, you must calculate the bandwidth manually by using the feedback divider M (ranging from 1 to 2^{16}-1), Equation 1, and Equation 2. The PLL is automatically stable in such case. In normal applications, APLLCTRL[BANDDIRECT] and SPLLCTRL[BANDDIRECT] must be 0; in this case, the bandwidth changes as a function of M.

$$A = floor\,(M \div 4) + 1$$

$$SELP = \begin{cases} A & if\ A < 31 \\ 31 & if\ A \geq 31 \end{cases}$$

Equation 1. SELP

$$A = \begin{cases} 1 & if\ M \geq 8000 \\ floor\,(8000 \div M) & if\ 8000 > M \geq 122 \\ 2 \times floor\,(M \div 4) + 3 & if\ 122 > M \geq 1 \end{cases}$$

$$SELI = \begin{cases} A & if\ A < 63 \\ 63 & if\ A \geq 63 \end{cases}$$

Equation 2. SELI

SELR must be 0.

NOTE

In some applications, you may prefer to change the bandwidth directly on the PLL. In such an application, you must write 1 to APLLCTRL[BANDDIRECT] or SPLLCTRL[BANDDIRECT].

34.3.4.4.3.2 Input clock

A good-quality input clock is necessary for good PLL performance. The input clock on clkin, optionally divided by the predivider ratio N, gives the reference frequency F_{ref} for the PLL loop (F_{ref} = F_{in} / N).

Use SOURCE[1:0] to select one of two inputs to the PLL: SOSC and FIRC. The input clock must have an accurate frequency before powering up the PLL. You can confirm a valid clock for SOSC and FIRC by using SOSCCSR[SOSCVLD] and FIRCCSR[FIRCACC], respectively.

For an optimal performance, use a reference frequency within the frequency range 5 MHz - 50 MHz. The total possible reference frequency range is specified in the data sheet.

34.3.4.4.3.3 Selecting divider ratios

The PLL contains three dividers: predivider (N), feedback-divider (M), and postdivider (P). The divider ratios can be selected in different ways.

Start up

For a correct start up, a low to high transition of PLLPWREN is required after the supplies are stable. The divider ratio data XDIV[n:0] (in which X is N, M, or P) must also be stable (see Figure 166) before PLLPWREN is made high.

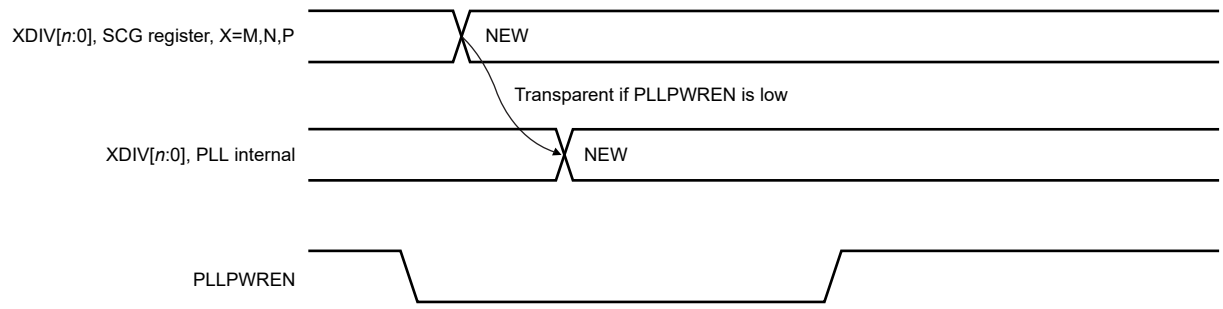


Figure 166. Loading of the divider ratios at power down

Changing divider ratio after starting up

There are two ways to change the divider ratio after the PLL start up: you can reset the PLL or you can read the divider ratio asynchronous.

When changing divider ratios, you must avoid forbidden divider ratio combinations at all times in order to keep the CCO frequency F_{cco} within the specified range. See the product data sheet for the range information. If you need to change more than one divider ratio, it is preferable to use the reset method.

Resetting the PLL

You can select the new divider ratios by making PLLPWREN low. After power down, the new divider ratios at the input buses XDIV[n:0] (in which X is N, M, or P) are loaded into the dividers (Figure 166). You must ensure that the divider ratio data XDIV[n:0] is stable before the PLLPWREN is made high (PLLPWREN=1). This can be done by resetting the PLL (PLLPWREN=0) until the divider ratio data XDIV[n:0] is stable.

Reading the divider ratio asynchronous

You can select the divider ratio of the different dividers (M, N, P) on-the-fly with the help of a handshake protocol (Figure 167). First, a new divider ratio has to be put at the divider input bus (XDIV), and after that a request (XREQ=1) must be given. As soon as the divider gives acknowledgment (XACK=1), the new divider ratio is loaded and the request can be made low (XREQ=0).

For a fractional application, write 1 to SEL_SS_MDIV. Use SS_MDIV_REQ and SS_MDIV[32:0].

NOTE

If the N divider is changed, you should also update the LOCK_TIME according to the new N.

NOTE

Updating divider ratio will clear PLL_LOCK until t_{pon} (defined in LOCK_TIME) is passed. Since PLL_LOCK=0 will gate off the PLL clock before on-the-fly update, you should switch the system to another clock source.

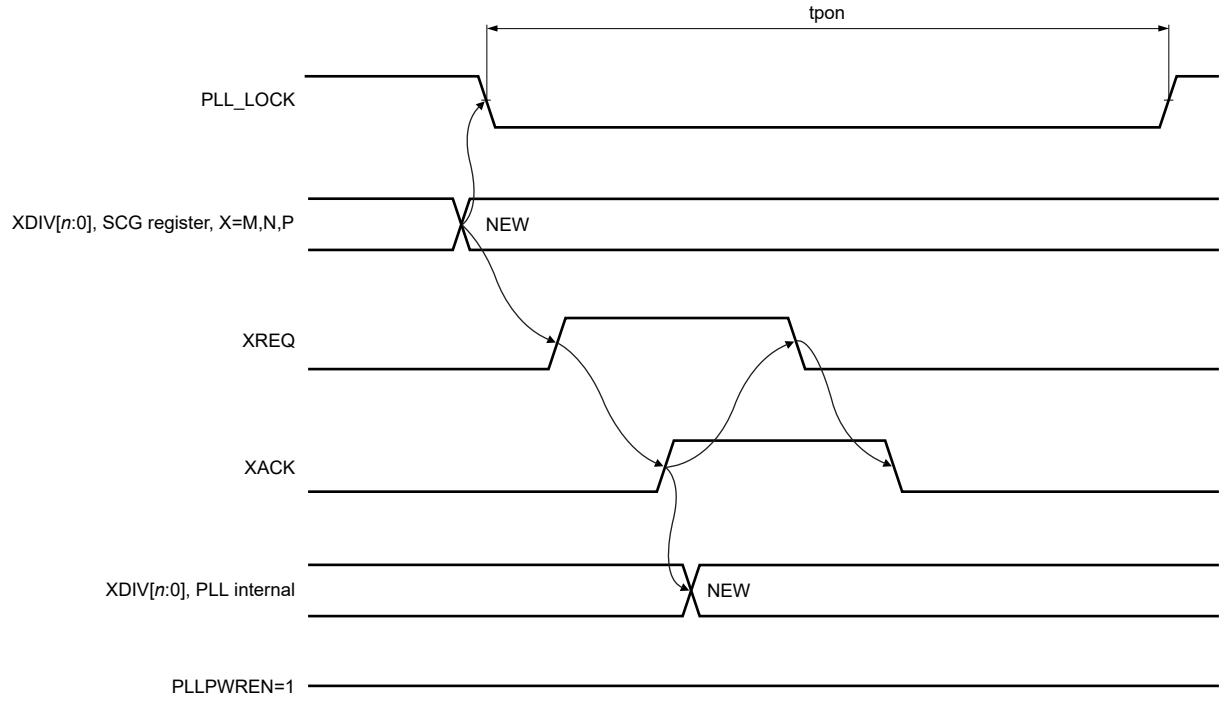


Figure 167. Handshake protocol for asynchronous update divider ratio

34.3.4.4.3.4 Mode 1: Normal operating mode

Typical operation of the PLL includes the following sequence to produce the PLL output:

1. An optional predivider of the PLL input.
2. A frequency multiplication.
3. An optional postdivider.

You can select the predivider and postdivider (see Table 327) to produce:

- Mode 1a: Normal operating mode without postdivider and without predivider
- Mode 1b: Normal operating mode with postdivider and without predivider
- Mode 1c: Normal operating mode without postdivider and with predivider
- Mode 1d: Normal operating mode with postdivider and with predivider

To produce the best phase-noise and jitter performance at the output of the PLL, you must use the highest possible reference clock (clkref) at the PFD. Therefore, NXP recommends modes 1a and 1b, if you can make the right output frequency without a predivider.

By using the postdivider and the divide-by-2 divider, the divider ratio of the PLL clock output (clkout) is even when APLLCTRL[BYPASSPOSTDIV2] or SPLLCTRL[BYPASSPOSTDIV2] is 0. For uneven divider ratios, the divide-by-2 divider can be bypassed when the BYPASSPOSTDIV2 field is 1.

NOTE

If BYPASSPOSTDIV2 is 1, the duty cycle of clkout cannot be 50%.

Table 327. Modes within the normal operating mode

Mode 1	BYPASSPREDIV	BYPASSPOSTDIV
Mode 1a	1	1
Mode 1b	1	0
Mode 1c	0	1
Mode 1d	0	0

Table 328 shows the terms that the equations in this section use.

Table 328. Mode 1*n* equation terms

Term	Description	Allowed values
M	Feedback divider	1 to 2^{16}-1
N	Predivider	1 to 2^{8}-1
P	Postdivider	1 to 2^{5}-1

Mode 1a: Normal operating mode without postdivider and without predivider

Mode 1a bypasses the postdivider and predivider. Equation 3 shows the operating frequency:

$$F_{out} = F_{CCO} = M \times F_{in}$$

Equation 3. Operating frequency in mode 1a

Mode 1b: Normal operating mode with postdivider and without predivider

Mode 1b bypasses the predivider. Equation 4 shows the operating frequency:

$$F_{out} = \frac{F_{cco}}{2 \times P} = \frac{M}{2 \times P} \times F_{in}$$

Equation 4. Operating frequency in mode 1b

Mode 1c: Normal operating mode without postdivider and with predivider

Mode 1c bypasses the postdivider with divide-by-2. Equation 5 shows the operating frequency:

$$F_{out} = F_{cco} = \frac{M}{N} \times F_{in}$$

Equation 5. Operating frequency in mode 1c

Mode 1d: Normal operating mode with postdivider and with predivider

Mode 1d does not bypass either divider. Equation 6 shows the operating frequency:

$$F_{out} = \frac{F_{cco}}{2 \times P} = \frac{M}{N \times 2 \times P} \times F_{in}$$

Equation 6. Operating frequency in mode 1d

34.3.4.4.3.5 Mode 3: Power down mode (PLLPWREN)

If the PLL is not used, or if it is turned off in a running application, power can be saved by putting the PLL in power down mode (PLLPWREN=0). Before this is done, the CPU and any peripherals that are not meant to be stopped must be running from a different clock source.

In power down mode, the oscillator stops and the output clkout will be low.

While in PLL power down mode, the PLL_LOCK will be low to indicate that the PLL is not in lock. When the PLL power down mode is terminated by setting the PLLPWREN field to one, the PLL will resume its normal operation and will make the PLL_LOCK high after the start up time.

34.3.4.4.3.6 Mode 5: Free running mode (FRM)

In normal applications, the output clock of the PLL quickly disappears if the input clock disappears. To avoid the fast disappearance of the clock, you can use free running mode.

In FRM mode, the PLL monitors the input clock clkin. If the input clock disappears for more than six periods, the CCO of the PLL starts running free. The output frequency will be within +1% and -20% for at least 200 µs. After 200 µs of self-clocking, the frequency keeps going down.

NOTE

If the PLL is in free running mode, the FRMDET flag is high.

NOTE

Free running mode only operates for reference frequencies above 100 kHz (F_{in} / N = F_{ref} > 100 kHz). To use the free running mode for 32 kHz <= F_{in} / N = F_{ref} < 100 kHz, write 1 to FRM_CLOCKSTABLE after the PLL output frequency is stable (after startup time t_{pon}).

Free running mode is designed so that if the input clock occurs again, the PLL goes out of free running and locks. Because the output clock has drifted away, overshoot and lock time have to be taken into account. In free running mode (FRM=1), you can choose the same divider settings as set by modes 1, 2, and 3 with logically FRM=1.

NOTE

When FRMDET=1 and the PLL clock monitor is enabled (PLLCM=1), the PLLERR flag is set. When PLL_ERROR=1, switch to another clock and clear the field. PLL_LOCK is not reliable in this case. After t_{pon}, if PLL_ERROR is 0, this means the PLL can be used again.

34.3.4.4.3.7 Mode 8: Enable mode (PLLCLKEN)

In Enable mode, the output clock (clkout) of the PLL is enabled. If PLLCLKEN=0, the output of the PLL is low (clkout=0). Ensure that no spikes occur at the output of the PLL (clkout) by switching into and out of the Enable mode.

Going out of Enable mode can be used for safely switching to other modes, such as power down mode and direct output (BYPASSPOSTDIV). This method ensures that no spikes can occur at the output.

34.3.4.5 PLL spread spectrum and fractional modes

The SSCG wrapper is a ΣΔ modulator for the multi-purpose PLL and makes it a fractional-N PLL.

By controlling the feedback divider of the PLL, the wrapper can also generate a spread spectrum clock. SSCG decreases electromagnetic interferences (EMI).

34.3.4.5.1 Functional features

- Handshake circuit to read the asynchronous SS_MDIV[32:0] at the wrapper
- Twenty-five fractional bits
- Preferred reference frequency F_{ref} > 3 MHz

- Center-spread modulation (default)
- Modulation waveform control
- Modulation waveform dithering
- Power-down mode (SS_PD)
- Triangular-wave modulation frequency:

$$fm = \frac{F_{ref}}{N_{ss}}$$

with N_{ss} ranging from 16 to 512

Equation 7. Triangular-wave modulation frequency

- Frequency-modulation depth:

$$\delta fmod_{pk\text{-}pk} = \frac{F_{ref} \times k_{ss}}{F_{cco}}$$

with k_{ss} ranging from 0 to 4

Equation 8. Frequency-modulation depth

34.3.4.5.2 Functional description

34.3.4.5.2.1 Basic functions

The SSCG WRAPPER is a second order ΣΔ modulator for the multipurpose PLL and makes it a fractional-N PLL. By controlling the feedback divider of the PLL, the wrapper can also generate a spread spectrum clock (Spread Spectrum Clock Generator).

The main block is the ΣΔ modulator which generates the appropriate MDIV[15:0] values from the input word SS_MDIV[32:0] to have at the output (clkout) of the PLL the correct frequency.

34.3.4.5.2.2 Main blocks

The SSCG WRAPPER consist of:

- A handshake circuit to read a new input word on the buses: SS_MDIV[32:0], MR[2:0], MF[2:0], MC[2:0]
- A waveform generator
- A second order ΣΔmodulator
- A multiplexer to bypass the wrapper

The SSCG WRAPPER will generate by default a center spread triangular waveform. It is also possible to generate a down spread triangular waveform which is explained Center spread versus down spread.

34.3.4.5.2.3 Handshake circuit

A new (asynchronous) input word (SS_MDIV[32:0], MF[2:0], MR[2:0], and MC[1:0]) can be read with the help of a handshake protocol (Figure 168).

First the new word must be put on the bus. Then a request (SS_MDIV_REQ=1) must be given. As soon as the acknowledge pin goes high (SS_MDIV_ACK=1), the new input word is loaded and the request can be made low (SS_MDIV_REQ=0).

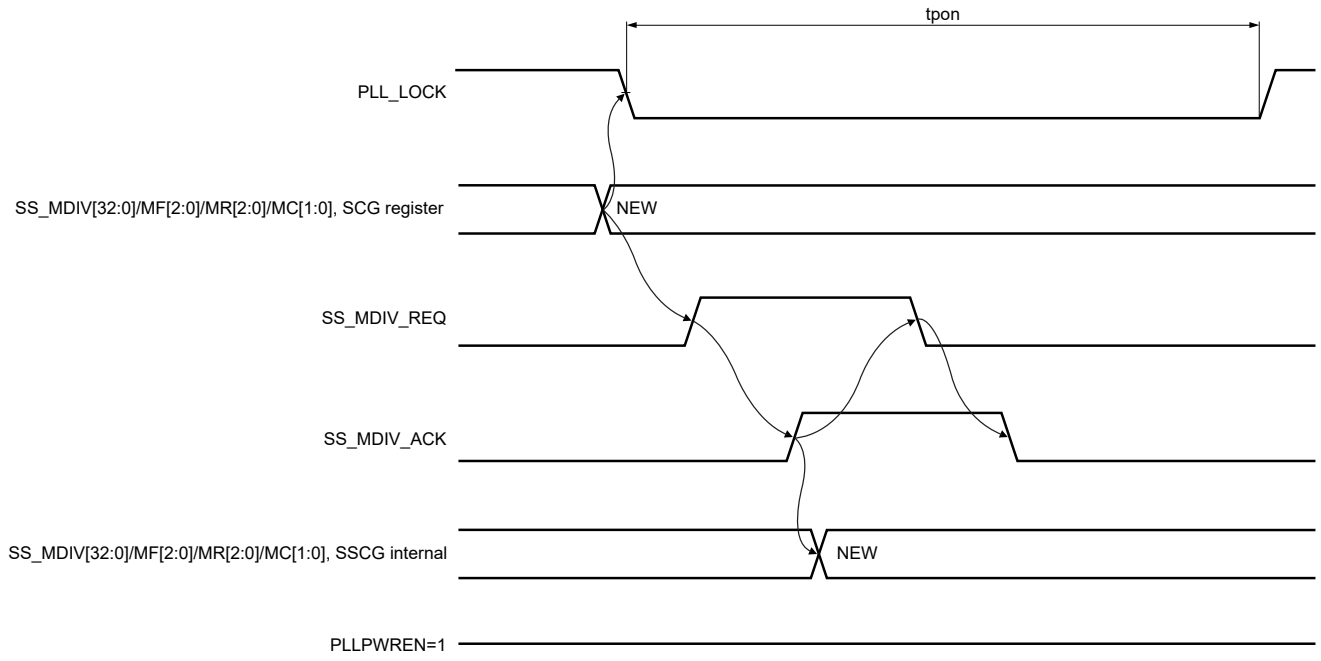


Figure 168. Handshake protocol of the input words SS_MDIV[32:0], MF[2:0], MR[2:0] and MC[1:0]

NOTE

Updating the divider ratio will clear the PLL_LOCK until t_{pon} (defined in LOCK_TIME) is passed. Because PLL_LOCK=0 will gate off the PLL clock, system should be switched to another clock source before on-the-fly updates.

34.3.4.5.2.4 Center spread versus down spread

By default, the wrapper generates a triangular waveform with center spread. This means that the average CCO frequency between MR[2:0] = 0 (no spread spectrum) and MR[2:0] > 0 is the same. see Equation 9.

$$M_{average,\ center\ spread} = M_{average}$$

Equation 9. Divider equivalence

Some applications require a triangular waveform with down spread. For down spread, instead of the average CCO frequency being the same, the maximum frequency must be the same in spread spectrum mode (MR[2:0]>0). For down spread, the average feedback divider ratio must be decreased depending on the modulation depth. See Equation 10 and Equation 11.

$$M_{average,\ down\ spread} = M_{average} - 0.25 \times k_{ss}$$

Equation 10. Average divider, down spread

$$M_{average} = 64.5,\ k_{ss} = 4$$

$$M_{average,center\ spread} = M_{average} = 64.5\ (SS_{MDIV[32:0]} = 0_1000_0001_0000_0000_0000_0000_0000_0000b = 0_8100_0000h)$$

$$M_{average,down\ spread} = M_{average} - 1 = 63.5\ (SS_{MDIV[32:0]} = 0_0111_1111_1000_0000_0000_0000_0000_0000b = 0_7F80_0000h)$$

Equation 11. Average divider example

34.3.4.5.2.5 Triangular wave modulation

You can calculate the clock cycle displacement and spectral tone reduction ΔP for the center spread triangular waveform modulation with a modulation frequency depth δfmodpk-pk and a modulation frequency fm.

Equation 12 shows the theoretical maximum clock-cycle displacement (peak-to-peak).

$$\Delta n_{max;theoretical} = \begin{cases} \frac{N_{ss} \times k_{ss}}{16} & if\ PLLCTRL[BYPASSPOSTDIV] = 1 \\ \frac{N_{ss} \times k_{ss}}{32 \times P_{PLL}} & if\ PLLCTRL[BYPASSPOSTDIV] = 0\ and\ PLLCTRL[PDIV] = 1 \end{cases}$$

Equation 12. Theoretical maximum clock-cycle displacement

In practice the clock cycle displacement could be larger. So, for safety reasons (buffer overflow) use Equation 13:

$$\Delta n_{max;practical} = \begin{cases} \frac{N_{ss} \times k_{ss}}{8} & if\ PLLCTRL[BYPASSPOSTDIV] = 1 \\ \frac{N_{ss} \times k_{ss}}{16 \times P_{PLL}} & if\ PLLCTRL[BYPASSPOSTDIV] = 0\ and\ PLLCTRL[PDIV] = 1 \end{cases}$$

Equation 13. Practical maximum clock-cycle displacement

Equation 14 shows the spectral tone reduction / EMI reduction ΔP at F_{out}.

$$\Delta P \approx \begin{cases} 10 \times \log\left(\frac{N_{ss} \times k_{ss}}{2}\right) & if\ PLLCTRL[BYPASSPOSTDIV] = 1 \\ 10 \times \log\left(\frac{N_{ss} \times k_{ss}}{4}\right) & if\ PLLCTRL[BYPASSPOSTDIV] = 0\ and\ PLLCTRL[PDIV] = 1 \end{cases}$$

Equation 14. ΔP at F_{out}

Table 329 shows the spectral tone reduction and clock cycle displacement for BYPASSPOSTDIV=0 and P_{PLL}=1.

Table 329. Values for different settings, BYPASSPOSTDIV = 0, P_{PLL} = 1

Table values are: ΔP Δn_{max}	MF[2:0]=000 N_{ss} = 512	MF[2:0]=001 N_{ss} ≈ 384	MF[2:0]=010 N_{ss} = 256	MF[2:0]=011 N_{ss} = 128	MF[2:0]=100 N_{ss} = 64	MF[2:0]=101 N_{ss} = 32	MF[2:0]=110 N_{ss} ≈ 24	MF[2:0]=111 N_{ss} = 16
MR[2:0]=000, K_{ss}≈0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0
MR[2:0]=001, K_{ss}≈0.5	21 dB 32	20 dB 24	18 dB 16	15 dB 8	12 dB 4	9 dB 2	8 dB 1.5	6 dB 1
MR[2:0]=010, K_{ss}≈0.75	23 dB 48	22 dB 32	20 dB 24	17 dB 12	14 dB 6	11 dB 3	10 dB 2.2	8 dB 1.5
MR[2:0]=011, K_{ss}≈1	24 dB 64	23 dB 48	21 dB 32	18 dB 16	15 dB 8	12 dB 4	11 dB 3	9 dB 2
MR[2:0]=100, K_{ss}≈1.5	26 dB 96	25 dB 64	25 dB 48	20 dB 24	17 dB 12	14 dB 6	13 dB 4.5	12 dB 4

Table continues on the next page...

Table 329. Values for different settings, BYPASSPOSTDIV = 0, P_{PLL} = 1 (continued)

Table values are: ΔP Δn_{max}	MF[2:0]=000 N_{ss} = 512	MF[2:0]=001 N_{ss} ≈ 384	MF[2:0]=010 N_{ss} = 256	MF[2:0]=011 N_{ss} = 128	MF[2:0]=100 N_{ss} = 64	MF[2:0]=101 N_{ss} = 32	MF[2:0]=110 N_{ss} ≈ 24	MF[2:0]=111 N_{ss} = 16
MR[2:0]=101, K_{ss}≈2	27 dB 128	26 dB 96	24 dB 64	21 dB 32	18 dB 16	15 dB 8	14 dB 6	12 dB 4
MR[2:0]=110, K_{ss}≈3	28 dB 192	28 dB 128	26 dB 96	23 dB 48	20 dB 24	17 dB 12	16 dB 9	14 dB 6
MR[2:0]=111, K_{ss}≈4	30 dB 256	29 dB 192	27 dB 128	24 dB 64	21 dB 32	18 dB 16	17 dB 12	15 dB 8

34.3.4.5.2.6 $M_{divider}$ value, stepsize, and output frequency

Table 330 shows examples of the average feedback-divider ratio $M_{average}$ (which defines) and stepsize.

$$M_{average} = \text{SS_MDIV}[32:25]_{dec} + 2^{-25} \times SS_MDIV[24:0]_{dec} + DITHER \times 2^{-26} = 2^{-25} \times SS_MDIV_{dec} + DITHER \times 2^{-26}$$

Equation 15. Average feedback-divider ratio

If the SSCG WRAPPER is in power down (SS_PD = 1), then the input word SS_MDIV[32:0] is not modulated. Examples of the feedback-divider ratio M and stepsize for SS_PD = 1 can be found in Table 331.

Table 330. Examples of $M_{average}$ and stepsize if SS_PD = 0

SS_MDIV[32:0]	$M_{average}$(DITHER=1)	$M_{average}$(DITHER=0)	Stepsize
0 1000 0000 0000 0000 0000 0000 0000 0000bin = 080000000hex = 64 *2^{25}	64 + 2^{-26}	64	1 ÷ (2^{25}*$M_{average}$) ≈ 0.48ppb
0 1000 0010 0000 0000 0000 0000 0000 0000bin = 082000000hex = 65 *2^{25}	65 + 2^{-26}	65	1 ÷ (2^{25}*$M_{average}$) ≈ 0.46ppb
0 1000 0010 0000 0000 0000 0000 0000 0001bin = 082000001hex = 65 *2^{25}+1	65 + 2^{-25} + 2^{-26}	65 + 2^{-25}	1 ÷ (2^{25}*$M_{average}$) ≈ 0.46ppb
0 1000 0011 0000 0000 0000 0000 0000 0000bin = 083000000hex = 65 * 2^{25}+2^{24}	65 + 2^{-1} + 2^{-26}	65 + 2^{-1}	1 ÷ (2^{25}*$M_{average}$) ≈ 0.46ppb
0 1011 1000 0000 0000 0000 0000 0000 0000bin = 0B8000000hex = 92 *2^{25}	92 + 2^{-26}	92	1 ÷ (2^{25}*$M_{average}$) ≈ 0.33ppb[1]
0 0011 0110 0000 0000 0000 0000 0000 0000bin = 036000000hex = 27 *2^{25}	27 + 2^{-26}	27	1 ÷ (2^{25}*$M_{average}$) ≈ 1.11ppb[2]

1. maximum $M_{average}$, minimum stepsize
2. minimum $M_{average}$, maximum stepsize

Table 331. Examples M and stepsize if SS_PD = 1

SS_MDIV[32:0]	M [1](SS_PD = 1)	stepsize[2](SS_PD = 1)
0 1000 0000 0000 0000 0000 0000 0000 0000bin = 080000000hex = 64 * 2^{25}	64	1 ÷ M ≈ 1.56%
0 1000 0010 0000 0000 0000 0000 0000 0000bin = 082000000hex = 65 *2^{25}	65	1 ÷ M ≈ 1.54%
0 1000 0010 0000 0000 0000 0000 0000 0001bin = 082000001hex = 65 *2^{25} + 1	65	1 ÷ M ≈ 1.54%
0 1000 0011 0000 0000 0000 0000 0000 0000bin = 083000000hex = 65 * 2^{25} + 2^{24}	65	1 ÷ M ≈ 1.54%
0 1011 1000 0000 0000 0000 0000 0000 0000bin = 0B8000000hex = 92 *2^{25}	92	1 ÷ M ≈ 1.09%
0 0011 0110 0000 0000 0000 0000 0000 0000bin = 036000000hex = 27 *2^{25}	27	1 ÷ M ≈ 3.70%

1. M = $SS_MDIV[32:25]_{dec}$
2. stepsize = 1 ÷ M

34.3.4.5.3 Application information

In Figure 165 of the SSCG, WRAPPER and PLL are shown.

The spread spectrum clock can be disabled by setting MR[2:0] to zero. If the fractional part of SS_MDIV[32:0] is set to zero (SS_MDIV[24:0]=0) then the feedback divider ratio will have a constant value (no fractional behavior).

NOTE

If the spread spectrum mode is enabled, choose N to ensure $F_{in}/N > 3$ MHz. The spread spectrum mode cannot be used when F_{in} = 32 kHz.

34.3.4.5.3.1 PLL bandwidth settings

For spread spectrum and fractional application, it is important to set the PLL bandwidth in a preferred setting. The preferred settings for the PLL bandwidth at 3 MHz < F_{ref} < 5 MHz are:

- SELR[3:0]=b0100
- SELI[5:0]=b000100
- SELP[4:0]=b00011

34.3.4.5.3.2 Input clock

For Spread spectrum and fractional application, the preferred reference frequency is F_{ref} > 3 MHz.

34.3.4.6 Procedure for determining PLL settings

In general, the PLL configuration values are as follows:

1. Identify a desired PLL output frequency. This may depend on a specific interface frequency needed or on expected CPU performance requirements, and may be limited by system power availability.
2. Determine which clock source to use as the PLL input. This is based on the required power or accuracy, or by the potential to obtain the desired PLL output frequency.
3. Identify PLL settings to obtain the desired output from the selected input. The F_{cco} frequency must be either the actual desired output frequency, or the desired output frequency times 2 x P, where P is from 1 to 31. The F_{cco} frequency must

also be a multiple of the PLL reference frequency, which is either the PLL input or the PLL input divided by N (where N is from 1 to 255).

4. There are several ways to obtain the same PLL output frequency. PLL power depends on F_{cco} (a lower frequency uses less power) and the divider used. Bypassing the input or output divider, or bypassing both, saves power.
5. Ensure that the selected settings meet all the PLL requirements:
   - F_{in} is in the range of 32 kHz to 150 MHz.
   - F_{cco} is in the range of 275 MHz to 550 MHz.
   - F_{out} is in the range of 4.3 MHz to 2x Max CPU frequency.
   - The predivider is either bypassed, or N is in the range of 1 to 255.
   - The postdivider is either bypassed, or P is in the range of 1 to 31.
   - M is in the range of 1 to 65,535.

NOTE

Also note that the PLL startup time becomes longer as F_{ref} drops. PLL accuracy and jitter is better with higher values of F_{ref}.

34.3.4.7 PLL setup sequence

The sequence of initializing and connecting the PLL is as follows:

1. Enable PLL LDO.
2. Ensure that the PLL output is disconnected from any downstream functions.
3. Select a PLL input clock source. See SOURCE field in APLL Control Register (APLLCTRL) and SPLL Control Register (SPLLCTRL).
4. Set up the PLL dividers and mode settings. See APLL and SPLL registers in Memory map and register definition.
5. Wait for the input clock source to be valid. It is indicated by SOSCCSR[SOSCVLD] for SOSC and FIRCCSR[FIRCACC] for FIRC.
6. Power up PLL and write 1 to PLLPWREN and PLLCLKEN. The PLL clock is gated until PLL_LOCK=1.
7. Wait for the PLL output to stabilize, when PLL_LOCK=1. The start-up time is tpll_lock.
8. If the PLL is used to clock the CPU, you can change the CPU Clock Divider setting for the operation with the PLL, if needed. This must be complete before connecting the PLL.
9. Connect the PLL to whichever downstream function you are using it with. See the chip's Clocking chapter for more information on the clock divider structure.

34.3.5 Clocks

SCG has the following input clocks:

- Bus clock for register access
- External USB PLL (UPLL) and Real-Time oscillator (ROSC) clocks as one of the system clock sources

SCG has the following output clocks:

- Main clock
- System PLL clock (SPLL)
- Auxiliary PLL clock (APLL)
- Fast internal reference clock (FIRC)
  - 144 MHz and 48 MHz

- Slow internal reference clock (SIRC)
  - 12 MHz and 1 MHz, divided from SIRC
- System oscillator clock (SOSC)

34.3.6 Resets

These are the SCG resets:

- The global reset is fed into this module. The entire module is reset by the global reset.
- The clock monitor generates a reset in the following cases:
  - SOSCCSR[SOSCCM] = 1 & SOSCCSR[SOSCCMRE] = 1 & SOSCCSR[SOSCERR] = 1
  - ROSCCSR[ROSCCM] = 1 & ROSCCSR[ROSCCMRE] = 1 & ROSCCSR[ROSCERR] = 1
  - APLLCSR[APLLCM] = 1 & APLLCSR[APLLCMRE] = 1 & APLLCSR[APLLERR] = 1
  - SPLLCSR[SPLLCM] = 1 & SPLLCSR[SPLLCMRE] = 1 & SPLLCSR[SPLLERR] = 1
  - UPLLCSR[UPLLCM] = 1 & UPLLCSR[UPLLCMRE] = 1 & UPLLCSR[UPLLERR] = 1

On the assertion of any reset source, the FIRC 48 MHz clock is enabled or starts up if not already running.

34.3.7 Interrupts

- Clock monitor generates an interrupt when any of the following occurs:
  - SOSCCSR[SOSCCM] = 1 & SOSCCSR[SOSCCMRE] = 0 & SOSCCSR[SOSCERR] = 1
  - ROSCCSR[ROSCCM] = 1 & ROSCCSR[ROSCCMRE] = 0 & ROSCCSR[ROSCERR] = 1
  - APLLCSR[APLLCM] = 1 & APLLCSR[APLLCMRE] = 0 & APLLCSR[APLLERR] = 1
  - SPLLCSR[SPLLCM] = 1 & SPLLCSR[SPLLCMRE] = 0 & SPLLCSR[SPLLERR] = 1
  - UPLLCSR[UPLLCM] = 1 & UPLLCSR[UPLLCMRE] = 0 & UPLLCSR[UPLLERR] = 1
- Interrupt can also be generated when any of the following occurs:
  - SOSC clock source is not valid. SOSCCSR[SOSCVLD_IE] = 1 & SOSCCSR[SOSCVLD] = 1
  - FIRC clock source is not accurate. FIRCCSR[FIRCACC_IE] = 1 & FIRCCSR[FIRCACC] = 1
  - FIRC trimming error detected. FIRCCSR[FIRCERR_IE] = 1 & FIRCCSR[FIRCERR] = 1
  - SIRC trimming error detected. SIRCCSR[SIRCERR_IE] = 1 & SIRCCSR[SIRCERR] = 1
  - APLL is not powered or locked. APLLCSR[APLL_LOCK_IE] = 1 & APLLCSR[APLL_LOCK] = 1
  - SPLL is not powered or locked. SPLLCSR[SPLL_LOCK_IE] = 1 & SPLLCSR[SPLL_LOCK] = 1

34.4 External signals

Table 332. External signal descriptions

Signal	Description	Mode	I/O
XTAL	SOSC Crystal output pin	SOSC	O
EXTAL	SOSC Crystal input pin	SOSC	I
CLKOUT	SCG clock output pin	SCG	O

34.5 Initialization

Out of reset, the following events occur:

- FIRC is default enabled and FIRCCFG[RANGE] is reset to zero. The FIRC 48 MHz clock is selected as the system clock source.
- SIRC is default enabled.
- All other clock sources are disabled.

34.6 Application information

34.6.1 SOSC configuration example

1. Write 1 to LDOCSR[LDOEN] to enable LDO
2. Write 1 to SOSCCFG[EREFS] to select SOSC (internal crystal oscillator) source
3. Write 0 to SOSCCFG[RANGE] to configure SOSC range
4. Write 0 to SOSCCSR[LK] to unlock SOSCCSR
5. Write 1 to SOSCCSR[SOSCCM] to enable SOSC clock monitor
6. Write 1 to SOSCCSR[SOSCSTEN] to enable SOSC in Deep Sleep mode, if needed
7. Write 1 to SOSCCSR[SOSCEN] to enable SOSC
8. Read SOSCCSR[SOSCVLD] until it returns 1, indicating SOSC is valid
9. Read SOSCCSR[SOSCERR] to ensure it returns 0, indicating there is no error
10. Write 1 to RCCR[SCS] to switch main clock to SOSC
11. Read CSR[SCS] until it returns 1, indicating the switch is complete

34.6.2 SIRC configuration example

34.6.2.1 SIRC normal configuration example

1. Write 0 to SIRCCSR[LK] to unlock SIRCCSR
2. Write 1 to SIRCCSR[SIRCSTEN] to enable SIRC in Deep Sleep mode, if needed
3. Write 1 to SIRCCSR[SIRC_CLK_PERIPH_EN] to enable SIRC clock for peripheral use
4. Read SIRCCSR[SIRCVLD] until it returns 1, indicating SIRC is valid
5. Read SIRCCSR[SIRCERR] to ensure it returns 0
6. Write 2 to RCCR[SCS] to switch main clock to SIRC
7. Read CSR[SCS] until it returns 2, indicating the switch is complete

34.6.2.2 SIRC auto trim example

1. Follow steps 1 through 8 in SOSC configuration example
2. Write 2 to SIRCTCFG[TRIMSRC] to select SOSC as auto trim clock source
3. Write 39 to SIRCTCFG[TRIMDIV] to divide SOSC to 1 MHz
4. Write 0 to SIRCCSR[LK] to unlock SIRCCSR
5. Write 1 to SIRCCSR[SIRCTREN] to enable auto trim
6. Write 1 to SIRCCSR[SIRCTRUP] to enable update

7. Read SIRCCSR[SIRCVLD] until it returns 1, indicating SIRC is valid
8. Read SIRCCSR[SIRCERR] to ensure it returns 0
9. Read SIRCCSR]TRIM_LOCK] until it returns 1.

The auto trim process continues until it is disabled. The SIRCSTAT register shows the locked trim values. If there is an unlock after lock, SIRCCSR[SIRCERR] is set.

34.6.3 FIRC configuration example

34.6.3.1 FIRC normal configuration example

1. Write 1 to FIRCCFG[RANGE] to select 144 MHz
2. Write 0 to FIRCCSR[LK] to unlock FIRCCSR
3. Write 1 to FIRCCSR[FIRC_FCLK_PERIPH_EN] to enable FIRC 144 MHz clock for peripheral use, if needed
4. Write 1 to FIRCCSR[FIRC_SCLK_PERIPH_EN] to enable FIRC 48 MHz clock for peripheral use, if needed
5. Write 1 to FIRCCSR[FIRCSTEN] to enable FIRC clock source in Deep Sleep mode, if needed
6. Write 1 to FIRCCSR[FIRCEN] to enable FIRC
7. Read FIRCCSR[FIRCVLD] until it returns 1, indicating FIRC is valid
8. Read FIRCCSR[FIRCERR] to ensure it returns 0
9. Write 3 to RCCR[SCS] to switch main clock to FIRC
10. Read CSR[SCS] until it returns 3, indicating the switch is complete

34.6.3.2 FIRC auto trim example

1. sosc_clk_enable (); // Enable SOSC clock
2. wait_sosc_valid (); // Wait SOSC to be valid
3. Write 2 to FIRCTCFG[TRIMSRC] to select SOSC as auto trim clock source
4. Write 39 to FIRCTCFG[TRIMDIV] to divide SOSC to 1 MHz
5. Write 0 to FIRCCSR[LK] to unlock FIRCCSR
6. Write 1 to FIRCCSR[FIRCTREN] to enable auto trim
7. Write 1 to FIRCCSR[FIRCTRUP] to enable update
8. Read FIRCCSR[FIRCVLD] until it returns 1, indicating FIRC is valid
9. Read FIRCCSR[FIRCERR] to ensure it returns 0
10. Read FIRCCSR[TRIM_LOCK] until it returns 1

The auto trim process continues until it is disabled. The FIRCSTAT register shows the locked trim values. If there is an unlock after lock, FIRCCSR[FIRCERR] is set.

34.6.4 ROSC configuration example

1. Write 0 to ROSCCSR[LK] to unlock ROSCCSR
2. Write 1 to ROSCCSR[ROSCCM] to enable ROSC clock monitor, if needed
3. Read ROSCCSR[ROSCVLD] until it returns 1, indicating ROSC is valid
4. Read ROSCCSR[ROSCERR] to ensure it returns 0, indicating there is no error
5. Write 4 to RCCR[SCS] to switch main clock to ROSC

6. Read CSR[SCS] until it returns 4, indicating the switch is complete

34.6.5 APLL configuration example (SPLL is similar as APLL)

34.6.5.1 APLL outputs 200 MHz clock with normal function

1. Write 1 to LDOCSR[LDOEN] to enable LDO
2. Configure APLL reference clock source: SOSC
    a. Write 1 to SOSCCFG[EREFS] to select SOSC source
    b. Write 2 to SOSCCFG[RANGE] to configure SOSC range
    c. Write 0 to SOSCCSR[LK] to unlock SOSCCSR
    d. Write 1 to SOSCCSR[SOSCCM] to enable SOSC clock monitor
    e. Write 1 to SOSCCSR[SOSCEN] to enable SOSC
    f. Read SOSCCSR[SOSCVLD] returns 1, indicating SOSC is valid
    g. Read SOSCCSR[SOSCERR] to ensure it returns 0, indicating there is no error
3. Configure APLL:

NOTE

clk_out = clk_in/N*M/P (bypass post divider)

    a. Write 0 to APLLCTRL[SOURCE] to select SOSC as clock source
    b. Write 6 to APLLCTRL[SELP]. The value should refer to register SELP
    c. Write 13 to APLLCTRL[SELI]. The value should refer to register SELI
    d. Write 0 to APLLCTRL[SELR]. The value should refer to register SELR
    e. Write 2 to APLLNDIV[NDIV] to configure N divider
    f. Write 2 to APLLPDIV[PDIV] to configure P divider
    g. Write 20 to APLLMDIV[MDIV] to configure M divider
    h. Write 0 to APLLCTRL[BYPASSPREDIV] to use the N divider
    i. Write 0 to APLLCTRL[BYPASSPOSTDIV] to use the P divider
    j. Write 1 to APLLCTRL[BYPASSPOSTDIV2] to bypass P divider_2
    k. Write 1 to APLLCTRL[FRM] to enable free run mode, if needed
    l. Write 0 to APLLCSR[LK] to unlock APLLCSR
    m. Write 1 to APLLCSR[APLLCM] to enable clock monitor
    n. Write 5A5A0001h to the TRIM_LOCK register to unlock the APLLLOCK_CNFG register
    o. Write 10300 to APLLLOCK_CNFG[LOCK_TIME] to configure lock time of APLL stable. 10300 equals 500 µs/50 ns + 300, where 50 ns is the period of clk_ref (clk_in/N).
    p. Write 3 to APLLCSR[APLLCLKEN] and APLLCSR[APLLPWREN] to enable and power on the APLL clock
    q. Either read APLLCSR[APLL_LOCK] until it returns 1, indicating APLL is locked, or enable APLLCSR[APLL_LOCK_IE] to use the interrupt
    r. Read APLLCSR[APLLERR] to ensure it returns 0
4. Write 5 to RCCR[SCS] to switch main clock to APLL
5. Read CSR[SCS] until it returns 5, indicating the switch is complete

34.6.5.2 APLL outputs 150 MHz clock with spread-spectrum function from FIRC 48 MHz reference clock

1. Write 1 to LDOCSR[LDOEN] to enable LDO
2. Configure APLL reference clock source: FIRC 48 MHz
    a. Write 1 to FIRCCSR_FIRC_SCLK_PERIPH_EN] to select 48 MHz
    b. Write 0 to FIRCCSR[LK] to unlock FIRCCSR
    c. Write 1 to FIRCCSR[FIRCEN] to enable FIRC
    d. Read FIRCCSR[FIRCVLD] until it returns 1, indicating FIRC is valid
    e. Read FIRCCSR[FIRCERR] to ensure it returns 0
3. Configure APLL:

NOTE

clk_out = clk_in/N*M/P (bypass post divider)

    a. Write 1 to APLLCTRL[LIMUPOFF] to set to spectrum and fractional applications
    b. Write 1 to APLLCTRL[SOURCE] to select FIRC as clock source
    c. Write 2 to APLLCTRL[SELP]. The value should refer to register SELP.
    d. Write 5 to APLLCTRL[SELI]. The value should refer to register SELI.
    e. Write 0 to APLLCTRL[SELR]. The value should refer to register SELR.
    f. Write 1 to APLLNDIV[NDIV] to configure N divider
    g. Write 2 to APLLPDIV[PDIV] to configure P divider
    h. Write 0C800000h to APLLSSCG0[SS_MDIV_LSB] to configure M divider (M = 6.25)
    i. Write 0 to APLLSSCG1[SS_MDIV_MSB] to configure M divider (M = 6.25)
    j. Write 1 to APLLSSCG1[SEL_SS_MDIV]
    k. Write 0 to APLLSSCG1[DITHER]
    l. Write 2 to APLLSSCG1[MC]
    m. Write 0 to APLLSSCG1[MR]
    n. Write 0 to APLLSSCG1[MF]
    o. Write 0 to APLLCTRL[BYPASSPREDIV] to use the N divider
    p. Write 0 to APLLCTRL[BYPASSPOSTDIV] to use the P divider
    q. Write 1 to APLLCTRL[BYPASSPOSTDIV2] to bypass P divider_2
    r. Write 1 to APLLCTRL[FRM] to enable free run mode, if needed
    s. Write 0 to APLLCSR[LK] to unlock APLLCSR
    t. Write 1 to APLLCSR[APLLCM] to enable clock monitor
    u. Write 5A5A0001h to the TRIM_LOCK register to unlock the APLLLOCK_CNFG register
    v. Write 24300 to APLLLOCK_CNFG[LOCK_TIME] to configure lock time of APLL stable. 24300 equals 500 µs/ 20.8 ns+300, where 20.8 ns is the period of clk_ref (clk_in/N).
    w. Write 0 to APLLSSCG1[SS_PD] to power up SSCG
    x. Write 3 to APLLCSR[APLLCLKEN] and APLLCSR[APLLPWREN] to enable and power on PLL clock
    y. Either read APLLCSR[APLL_LOCK] until it returns 1, indicating APLL is locked, or enable APLLCSR_APLL_LOCK_IE to use the interrupt

z. Read APLLCSR[APLLERR] to ensure it returns 0

4. Write 5 to RCCR[SCS] to switch main clock to APLL
5. Read CSR[SCS until it returns 5, indicating the switch is complete

34.6.6 UPLL configuration example

1. Write 0 to UPLLCSR[LK] to unlock UPLLCSR
2. Write 1 to UPLLCSR[UPLLCM] to enable clock monitor
3. Read UPLLCSR[UPLLVLD] until it returns 1, indicating UPLL is valid
4. Read UPLLCSR[UPLLERR] to ensure it returns 0
5. Write 7 to RCCR[SCS] to switch main clock to UPLL
6. Read CSR[SCS] until it returns 7, indicating the switch is complete
7. Write 1 to UPLLCSR[LK] to lock UPLLCSR

34.7 Memory map and register definition

This section includes information about the memory map and register definitions:

- Accesses on non-implemented registers that are outside the SCG maximum illegal address range generate a transfer error.
- Read accesses on non-implemented registers that are inside the SCG maximum illegal address range do not generate a transfer error.
- Write accesses on non-implemented registers that are inside the SCG maximum illegal address range generate a transfer error.
- Write accesses on read-only registers generate a transfer error.
- Only 32-bit writes are allowed for any writable SCG registers. Writes of 8 or 16 bits result in transfer errors.

34.7.1 SCG register descriptions

34.7.1.1 SCG memory map

SCG0 base address: 4004_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	R	0000_0000h
4h	Parameter Register (PARAM)	32	R	0000_01FEh
8h	Trim Lock register (TRIM_LOCK)	32	RW	0000_0000h
10h	Clock Status Register (CSR)	32	R	0300_0000h
14h	Run Clock Control Register (RCCR)	32	RW	0300_0000h
100h	SOSC Control Status Register (SOSCCSR)	32	RW	0000_0000h
108h	SOSC Configuration Register (SOSCCFG)	32	RW	0000_0000h
200h	SIRC Control Status Register (SIRCCSR)	32	RW	0100_0020h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
20Ch	SIRC Trim Configuration Register (SIRCTCFG)	32	RW	0000_0000h
210h	SIRC Trim Register (SIRCTRIM)	32	RW	See section
218h	SIRC Auto-trimming Status Register (SIRCSTAT)	32	RW	See section
300h	FIRC Control Status Register (FIRCCSR)	32	RW	See section
308h	FIRC Configuration Register (FIRCCFG)	32	RW	0000_0000h
30Ch	FIRC Trim Configuration Register (FIRCTCFG)	32	RW	0000_0000h
310h	FIRC Trim Register (FIRCTRIM)	32	RW	See section
318h	FIRC Auto-trimming Status Register (FIRCSTAT)	32	RW	See section
400h	ROSC Control Status Register (ROSCCSR)	32	RW	0000_0000h
500h	APLL Control Status Register (APLLCSR)	32	RW	0000_0000h
504h	APLL Control Register (APLLCTRL)	32	RW	0000_0000h
508h	APLL Status Register (APLLSTAT)	32	R	0000_0000h
50Ch	APLL N Divider Register (APLLNDIV)	32	RW	0000_0001h
510h	APLL M Divider Register (APLLMDIV)	32	RW	0000_0001h
514h	APLL P Divider Register (APLLPDIV)	32	RW	0000_0001h
518h	APLL LOCK Configuration Register (APLLLOCK_CNFG)	32	RW	0000_4F4Ch
520h	APLL SSCG Status Register (APLLSSCGSTAT)	32	R	0000_0000h
524h	APLL Spread Spectrum Control 0 Register (APLLSSCG0)	32	RW	0000_0000h
528h	APLL Spread Spectrum Control 1 Register (APLLSSCG1)	32	RW	8000_0000h
5F4h	APLL Override Register (APLL_OVRD)	32	RW	0000_0000h
600h	SPLL Control Status Register (SPLLCSR)	32	RW	0000_0000h
604h	SPLL Control Register (SPLLCTRL)	32	RW	0000_0000h
608h	SPLL Status Register (SPLLSTAT)	32	R	0000_0000h
60Ch	SPLL N Divider Register (SPLLNDIV)	32	RW	0000_0001h
610h	SPLL M Divider Register (SPLLMDIV)	32	RW	0000_0001h
614h	SPLL P Divider Register (SPLLPDIV)	32	RW	0000_0001h
618h	SPLL LOCK Configuration Register (SPLLLOCK_CNFG)	32	RW	0000_4F4Ch
620h	SPLL SSCG Status Register (SPLLSSCGSTAT)	32	R	0000_0000h
624h	SPLL Spread Spectrum Control 0 Register (SPLLSSCG0)	32	RW	0000_0000h
628h	SPLL Spread Spectrum Control 1 Register (SPLLSSCG1)	32	RW	8000_0000h
6F4h	SPLL Override Register (SPLL_OVRD)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
700h	UPLL Control Status Register (UPLLCSR)	32	RW	0000_0000h
800h	LDO Control and Status Register (LDOCSR)	32	RW	0000_0008h

34.7.1.2 Version ID Register (VERID)

Offset

Register	Offset
VERID	0h

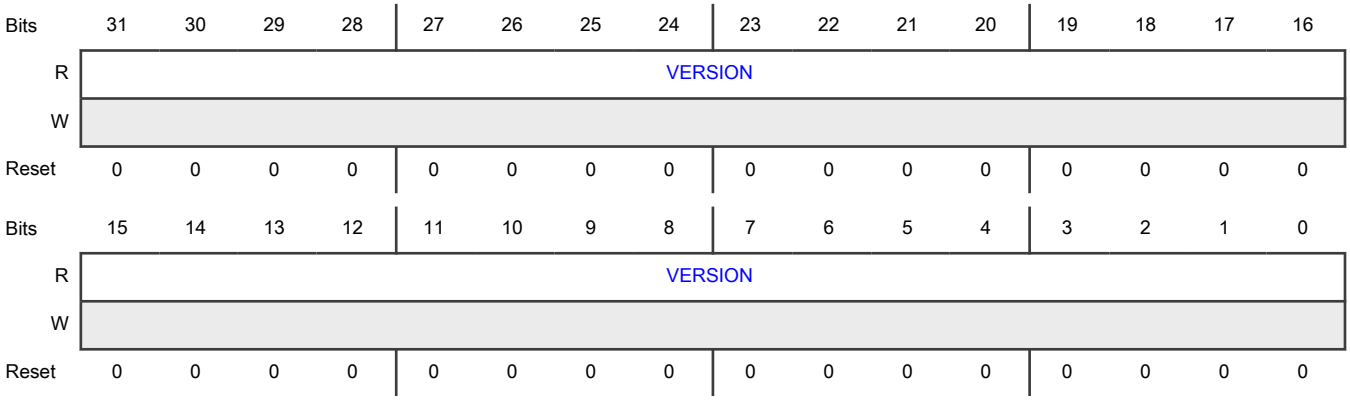
Function

Indicates the version integrated for this instance on the device.

NOTE

Writing to this register results in a transfer error.

Diagram



Fields

Field	Function
31-0 VERSION	SCG Version Number This read-only field returns the SCG module version number.

34.7.1.3 Parameter Register (PARAM)

Offset

Register	Offset
PARAM	4h

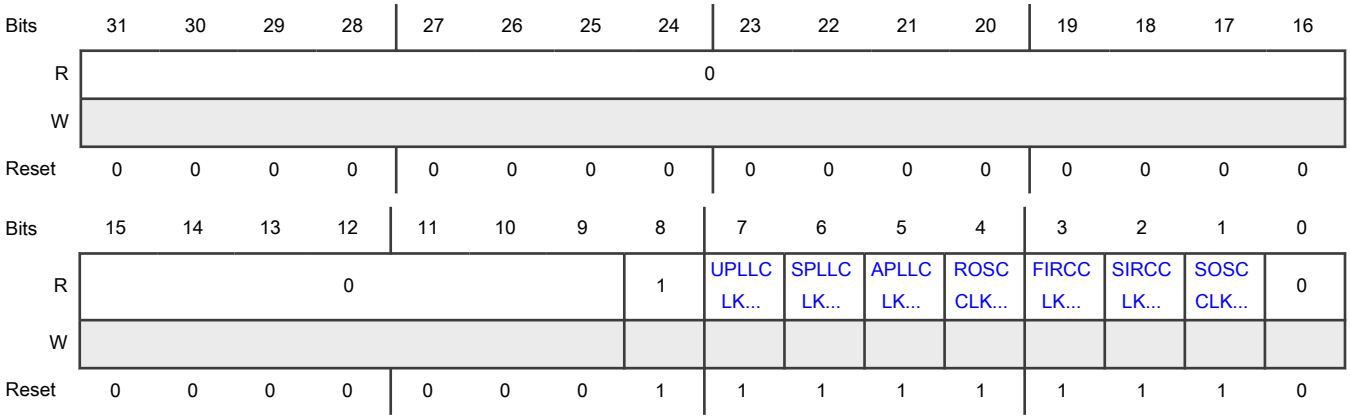
Function

Indicates the feature parameters for this instance on the device.

NOTE

Writing to this register results in a transfer error.

Diagram



Fields

Field	Function
31-9 —	Reserved
8 —	Reserved
7 UPLLCLKPRES	UPLL Clock Present Indicates that the UPLL clock source is present. 0b - UPLL clock source is not present 1b - UPLL clock source is present
6 SPLLCLKPRES	SPLL Clock Present Indicates that the SPLL clock source is present. 0b - SPLL clock source is not present

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - SPLL clock source is present
5 APLLCLKPRES	APLL Clock Present Indicates that the APLL clock source is present. 0b - APLL clock source is not present 1b - APLL clock source is present
4 ROSCCLKPRES	ROSC Clock Present Indicates that the ROSC clock source is present. 0b - ROSC clock source is not present 1b - ROSC clock source is present
3 FIRCCLKPRES	FIRC Clock Present Indicates that the FIRC clock source is present. 0b - FIRC clock source is not present 1b - FIRC clock source is present
2 SIRCCLKPRES	SIRC Clock Present Indicates that the SIRC clock source is present. 0b - SIRC clock source is not present 1b - SIRC clock source is present
1 SOSCCLKPRES	SOSC Clock Present Indicates that the SOSC clock source is present. 0b - SOSC clock source is not present 1b - SOSC clock source is present
0 —	Reserved

34.7.1.4 Trim Lock register (TRIM_LOCK)

Offset

Register	Offset
TRIM_LOCK	8h

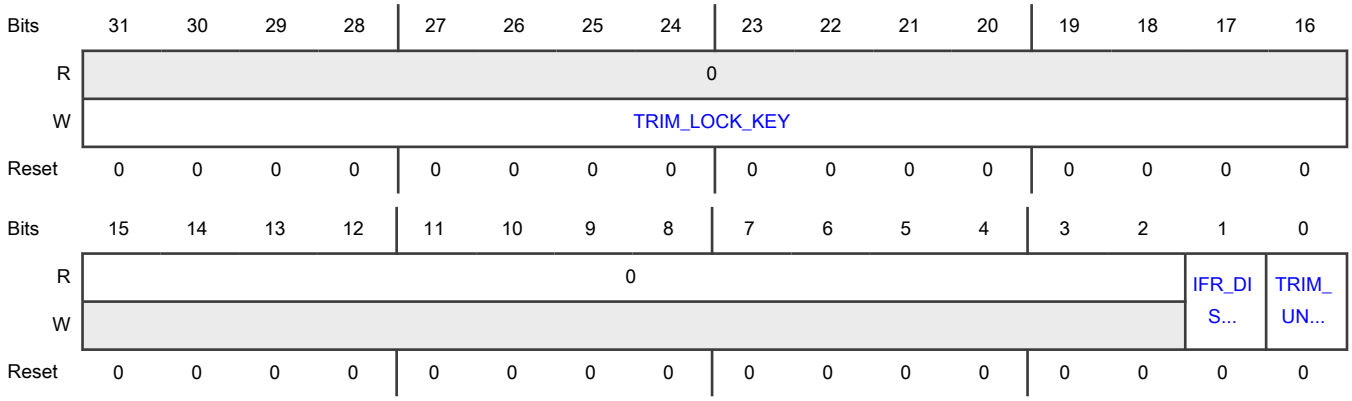
Function

Contains the TRIM_UNLOCK and IFR_DISABLE fields.

NOTE

These fields reset solely on a POR or LVD event.

Diagram



Fields

Field	Function
31-16 TRIM_LOCK_KEY	TRIM_LOCK_KEY Write value that is used to unlock writes to the IFR_DISABLE and TRIM_UNLOCK fields. Write 5A5Ah to allow writes to IFR_DISABLE and TRIM_UNLOCK to take effect. **NOTE** This field is writable but reads always return 0.
15-2 —	Reserved
1 IFR_DISABLE	IFR_DISABLE Locks IFR write access to SCG trim registers. When set, this field prevents SCG trim registers from loading IFR during warm reset. **NOTE** IFR_DISABLE is only writable during write access to TRIM_LOCK when the TRIM_LOCK[31:16] write value equals 16'h5a5a. 0b - IFR write access to SCG trim registers not disabled. The SCG Trim registers are reprogrammed with the IFR values after any system reset. 1b - IFR write access to SCG trim registers during system reset is blocked.
0 TRIM_UNLOCK	TRIM_UNLOCK Locks user write access to SCG Trim and PLL LOCK Configuration registers. **NOTE** TRIM_UNLOCK is only writable during write access to TRIM_LOCK when the TRIM_LOCK[31:16] write value equals 16'h5a5a.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - SCG Trim registers are locked and not writable. 1b - SCG Trim registers are unlocked and writable.

34.7.1.5 Clock Status Register (CSR)

Offset

Register	Offset
CSR	10h

Function

Shows the currently configured clock source that is generating the system clock.

NOTE

Writing to the Clock Status Register (CSR) results in a transfer error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				SCS				0							
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 SCS	System Clock Source Returns the currently configured clock source generating the system clock. Reports the configuration set by the RCCR. 0000b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0001b - SOSC 0010b - SIRC 0011b - FIRC 0100b - ROSC 0101b - APLL 0110b - SPLL 0111b - UPLL 1000b-1111b - Reserved
23-0 —	Reserved

34.7.1.6 Run Clock Control Register (RCCR)

Offset

Register	Offset
RCCR	14h

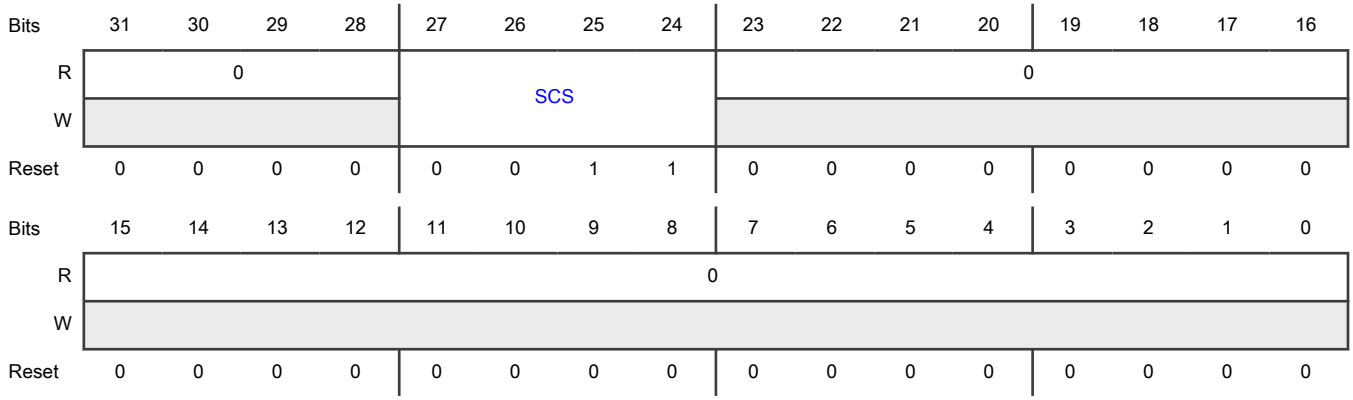
Function

Selects the clock source generating the system clock.

NOTE

Switching to a new system clock source must be done after previous RCCR changes are complete and the Clock Status Register has been updated to match the present RCCR setting.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 SCS	System Clock Source Selects the clock source generating the system clock in Run mode. Attempts to select a clock that is not valid are ignored. In Run mode, the clock source must be enabled and valid before system clocks can switch to a different clock source. **NOTE** You must wait for the previous SCS clock switch to complete before programming SCS to a different value. 0000b - Reserved 0001b - SOSC 0010b - SIRC 0011b - FIRC 0100b - ROSC 0101b - APLL 0110b - SPLL 0111b - UPLL 1000b-1111b - Reserved
23-0 —	Reserved

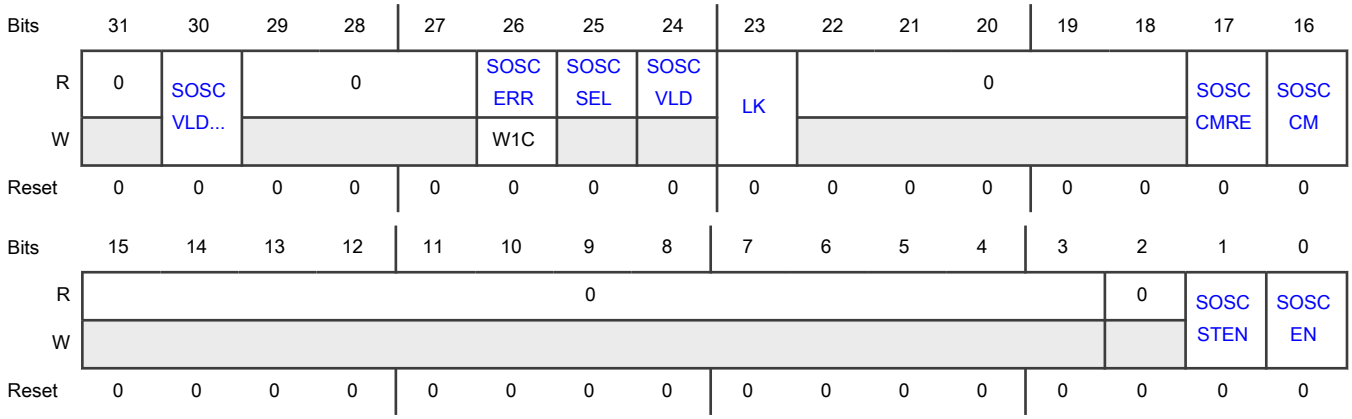
34.7.1.7 SOSC Control Status Register (SOSCCSR)

Offset

Register	Offset
SOSCCSR	100h

Function
Contains control and status fields for the SOSC clock source.

Diagram



Fields

Field	Function
31 —	Reserved
30 SOSCVLD_IE	SOSC Valid Interrupt Enable Generates an interrupt when SOSCVLD is asserted. 0b - SOSCVLD interrupt is not enabled 1b - SOSCVLD interrupt is enabled
29-27 —	Reserved
26 SOSCERR	SOSC Clock Error This flag is reset on Chip POR only. You can also clear this flag by writing 1. 0b - SOSC Clock Monitor is disabled or has not detected an error 1b - SOSC Clock Monitor is enabled and detected an error
25 SOSCSEL	SOSC Selected Indicates if the SOSC clock source is selected as the system clock source. 0b - SOSC is not the system clock source 1b - SOSC is the system clock source
24 SOSCVLD	SOSC Valid Indicates if the SOSC clock source is valid. 0b - SOSC is not enabled or clock is not valid 1b - SOSC is enabled and output clock is valid
23	Lock Register

Table continues on the next page...

Table continued from the previous page...

Field	Function
LK	Locks this register so that it cannot be written to. **NOTE** This field can be cleared or set at any time. 0b - This Control Status Register can be written 1b - This Control Status Register cannot be written
22-18 —	Reserved
17 SOSCCMRE	SOSC Clock Monitor Reset Enable Enables the SOSCERR generate reset. 0b - Clock monitor generates an interrupt when an error is detected 1b - Clock monitor generates a reset when an error is detected
16 SOSCCM	SOSC Clock Monitor Enable Enables the clock monitor when SOSCVLD is set. SOSC clock monitor remains enabled in Deep Sleep mode only if SOSCCM and SOSCSTEN are enabled. If the clock is disabled in Deep Sleep mode, then SOSCCM must be cleared to prevent unexpected SOSC loss of clock. The clock monitor is always disabled in Power Down and Deep Power Down modes. When the clock monitor is disabled in a low power mode, it remains disabled until the clock valid flag is set, following exit from that mode. **NOTE** The reference clock used to monitor the SOSC is the SIRC. This clock must be enabled in order to monitor the SOSC. SIRC is automatically disabled in Power Down and Deep Power Down modes and the SOSC clock monitor is disabled. 0b - SOSC Clock Monitor is disabled 1b - SOSC Clock Monitor is enabled
15-3 —	Reserved
2 —	Reserved
1 SOSCSTEN	SOSC Stop Enable Enables the SOSC clock source in Deep Sleep mode if SOSCEN is set. 0b - SOSC is disabled in Deep Sleep mode 1b - SOSC is enabled in Deep Sleep mode only if SOSCEN is set
0	SOSC Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
SOSCEN	Enables the SOSC clock source. 0b - SOSC is disabled 1b - SOSC is enabled

34.7.1.8 SOSC Configuration Register (SOSCCFG)

Offset

Register	Offset
SOSCCFG	108h

Function

Contains the clock frequency range select and external reference select for the SOSC clock source.

NOTE

You cannot change the SOSCCFG register when the SOSC is enabled. When the SOSC is enabled, writes to this register are ignored and there is no transfer error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0				0		RANGE		0	EREFS	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11-8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-6 —	Reserved
5-4 RANGE	SOSC Range Select Selects the frequency range for the system crystal oscillator (OSC). 00b - Frequency range select of 16-20 MHz. 01b - Frequency range select of 20-30 MHz. 10b - Frequency range select of 30-50 MHz. 11b - Frequency range select of 50-66 MHz.
3 —	Reserved
2 EREFS	External Reference Select Selects the source for the external reference clock. This field selects which clock is output from the SOSC into the SCG: either from the crystal oscillator or from an external clock input. 0b - External reference clock selected. LDO can be disabled in this case. 1b - Internal crystal oscillator of OSC selected.
1-0 —	Reserved

34.7.1.9 SIRC Control Status Register (SIRCCSR)

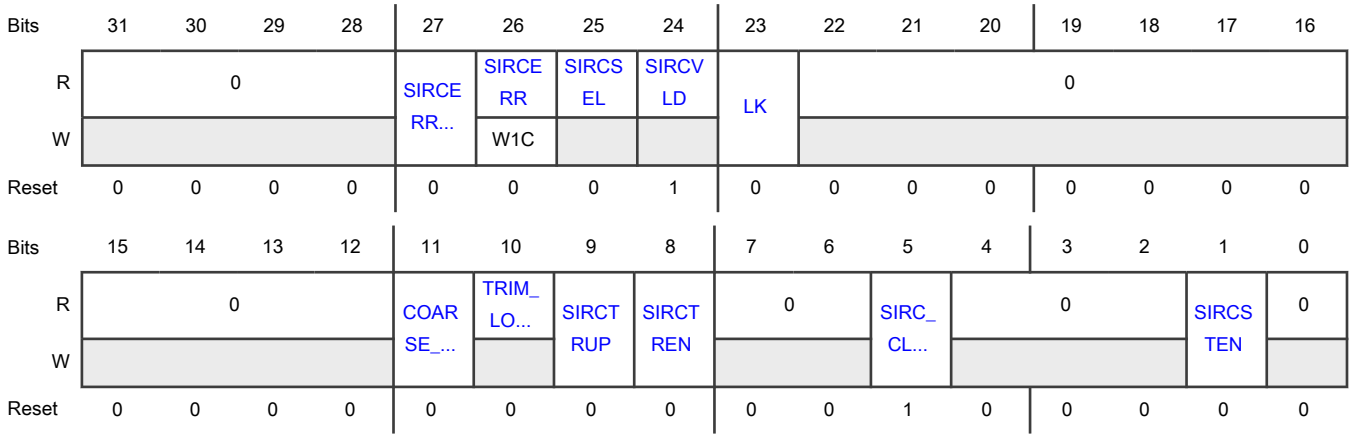
Offset

Register	Offset
SIRCCSR	200h

Function

Contains control and status bits for the SIRC clock source.

Diagram



Fields

Field	Function
31-28 —	Reserved
27 SIRCERR_IE	SIRC Clock Error Interrupt Enable Generates an interrupt when SIRCERR is asserted. 0b - SIRCERR interrupt is not enabled 1b - SIRCERR interrupt is enabled
26 SIRCERR	SIRC Clock Error This flag is reset on chip POR only. You can also clear this flag by writing 1. 0b - Error not detected with the SIRC trimming 1b - Error detected with the SIRC trimming
25 SIRCSEL	SIRC Selected Indicates if the SIRC clock source is selected as the system clock source. 0b - SIRC is not the system clock source 1b - SIRC is the system clock source
24 SIRCVLD	SIRC Valid Indicates if the SIRC clock source is valid. 0b - SIRC is not enabled or clock is not valid 1b - SIRC is enabled and output clock is valid
23 LK	Lock Register This field can be cleared or set at any time. Use this field to prevent runaway code from changing SIRC clock configurations.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Control Status Register can be written 1b - Control Status Register cannot be written
22-12 —	Reserved
11 COARSE_TRIM_BYPASS	Coarse Auto Trim Bypass Use this bit is to bypass the coarse trim step when auto-trimming. 0b - SIRC coarse auto-trim is not bypassed 1b - SIRC coarse auto-trim is bypassed
10 TRIM_LOCK	SIRC TRIM LOCK When SIRCTREN and SIRCTRUP are enabled, the TRIM_LOCK register indicates when auto trimming is complete and output SIRC frequency has locked to target SIRC range. **NOTE** • This field is automatically cleared if SIRCTREN and SIRCTRUP are not set. • TRIM_LOCK is not set when SIRCTCFG[TRIMSRC] = 2'b00, i.e., using full speed USB as the trim source. 0b - SIRC auto trim not locked to target frequency range 1b - SIRC auto trim locked to target frequency range
9 SIRCTRUP	SIRC Trim Update Allows the SIRCSTAT to be updated by auto-trimming hardware. 0b - Disables SIRC trimming updates 1b - Enables SIRC trimming updates
8 SIRCTREN	SIRC 12 MHz Trim Enable (SIRCCFG[RANGE]=1) Enables the auto trim of SIRC 12 MHz by an external clock source. 0b - Disables trimming SIRC to an external clock source 1b - Enables trimming SIRC to an external clock source
7-6 —	Reserved
5 SIRC_CLK_PERIPH_EN	SIRC Clock to Peripherals Enable Enables the SIRC clock for peripheral use. 0b - SIRC clock to peripherals is disabled 1b - SIRC clock to peripherals is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
4-2 —	Reserved
1 SIRCSTEN	SIRC Stop Enable Enables the SIRC clock source in specific power modes. See the chip-specific section for information on what conditions apply when SIRCSTEN is used. 0b - SIRC is disabled in Deep Sleep mode 1b - SIRC is enabled in Deep Sleep mode
0 —	Reserved

34.7.1.10 SIRC Trim Configuration Register (SIRCTCFG)

Offset

Register	Offset
SIRCTCFG	20Ch

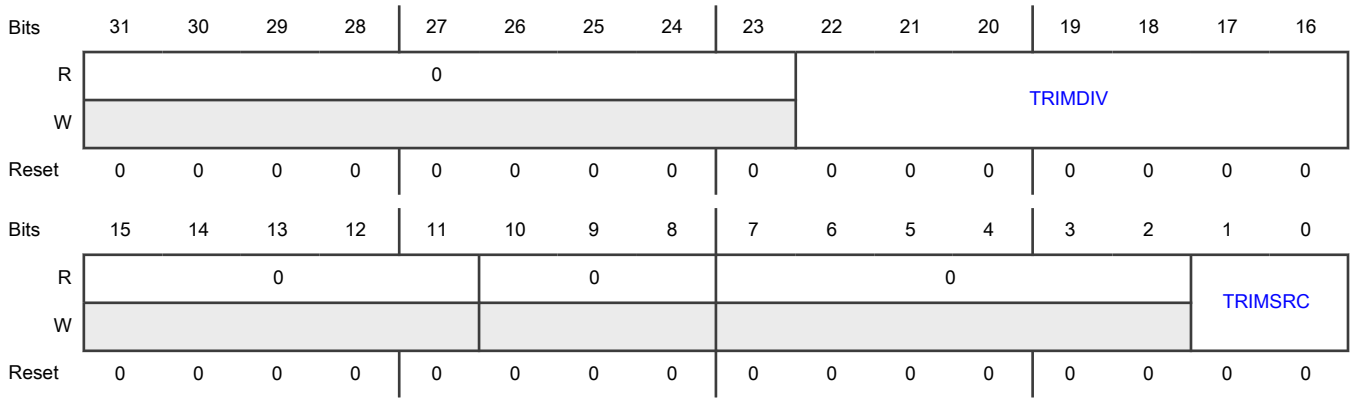
Function

Contains the auto trim clock source select and trim clock divider control for the SIRC clock source.

NOTE

The SIRCTCFG register cannot be changed when SIRC tuning is enabled. When the SIRC tuning is enabled, writes to this register are ignored and there is no transfer error.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-16 TRIMDIV	SIRC Trim Predivider Divider of SOSC for SIRC trimming. When selecting SOSC as the SIRC trimming source, the TRIMDIV register must be set to correct div ratio to generate 1 MHz output reference trimming clock. **NOTE** TRIMDIV is an N-Divider supporting a div ratio of 1 (TRIMDIV=00h) to a div ratio of 128 (TRIMDIV=7Fh).
15-11 —	Reserved
10-8 —	Reserved
7-2 —	Reserved
1-0 TRIMSRC	Trim Source Configures the external clock source to tune the SIRC. TRIMSRC must be configured before programming SIRCSTAT register for trim update. 00b - Reserved 01b - Reserved 10b - SOSC. This option requires that SOSC be divided using the TRIMDIV field to get a frequency of 1 MHz. 11b - ROSC (32.768 kHz)

34.7.1.11 SIRC Trim Register (SIRCTRIM)

Offset

Register	Offset
SIRCTRIM	210h

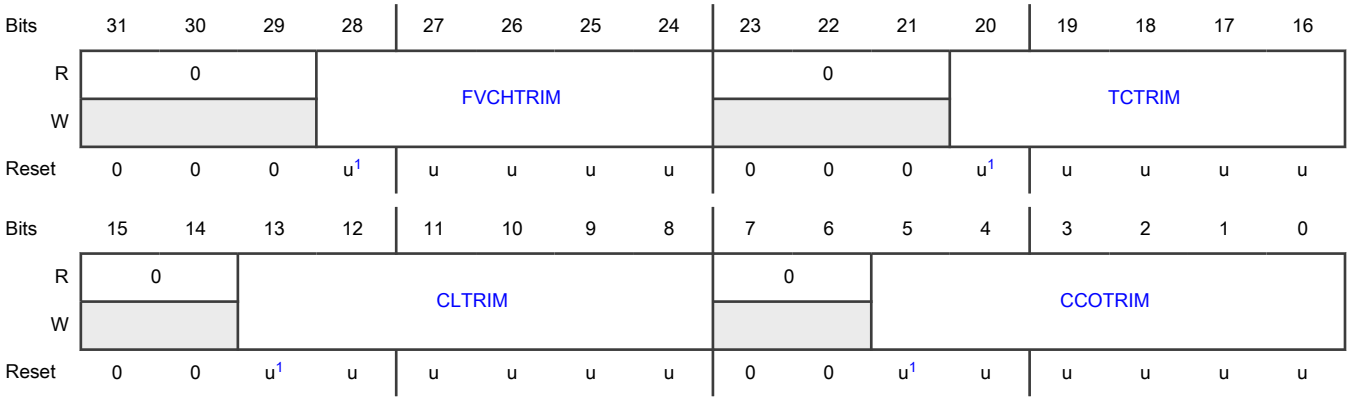
Function

Holds the trim values for the SIRC clock source. This register is loaded from IFR during reset.

NOTE

Writes to this register are protected by the TRIM_LOCK.

Diagram



1. Reset values are loaded out of IFR.

Fields

Field	Function
31-29 —	Reserved
28-24 FVCHTRIM	Calibrates the replica voltage in FSU for CCO to get well frequency at initial period
23-21 —	Reserved
20-16 TCTRIM	Trim Temp Trim bus to calibrate the relationship between freq and temperature.
15-14 —	Reserved
13-8 CLTRIM	CL Trim Trims bus to calibrate the freq of CCO in close loop mode to within approximately ±0.6% of the target 12 MHz frequency.
7-6 —	Reserved
5-0 CCOTRIM	CCO Trim Trims bus to calibrate the freq of CCO in open loop mode to within approximately ±2.5% of the target 12 MHz frequency.

34.7.1.12 SIRC Auto-trimming Status Register (SIRCSTAT)

Offset

Register	Offset
SIRCSTAT	218h

Function

This register is loaded from IFR during reset.

This register is uploaded with the trim values generated by SIRC auto-trimming. SIRC auto-trimming is enabled when SIRCTREN=1 and SIRCTRUP=1. When SIRCTREN=1 and SIRCTRUP=0 (write 10 or 11 to TRIMSRC), writes to this register are allowed and values written to this register are used to trim SIRC clock.

NOTE

You must program TRIMSRC to 10 or 11. Writes to this register are allowed and values written to this register are used to trim the SIRC clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		CLTRIM						0		CCOTRIM					
W																
Reset	0	0	u[1]	u	u	u	u	u	0	0	u[1]	u	u	u	u	u

1. Reset values are loaded out of IFR.

Fields

Field	Function
31-14 —	Reserved
13-8 CLTRIM	CL Trim Use CLTRIM to calibrate the frequency of CCO in close loop mode and to trim the SIRC clock to within approximately ±0.6% of the target 12 MHz frequency.
7-6 —	Reserved
5-0	CCO Trim

Table continues on the next page...

Table continued from the previous page...

Field	Function
CCOTRIM	Use CCOTRIM to calibrate the frequency of CCO in open loop mode and to coarsely trim the SIRC clock to within approximately ±2.5% of the target 12 MHz frequency.

34.7.1.13 FIRC Control Status Register (FIRCCSR)

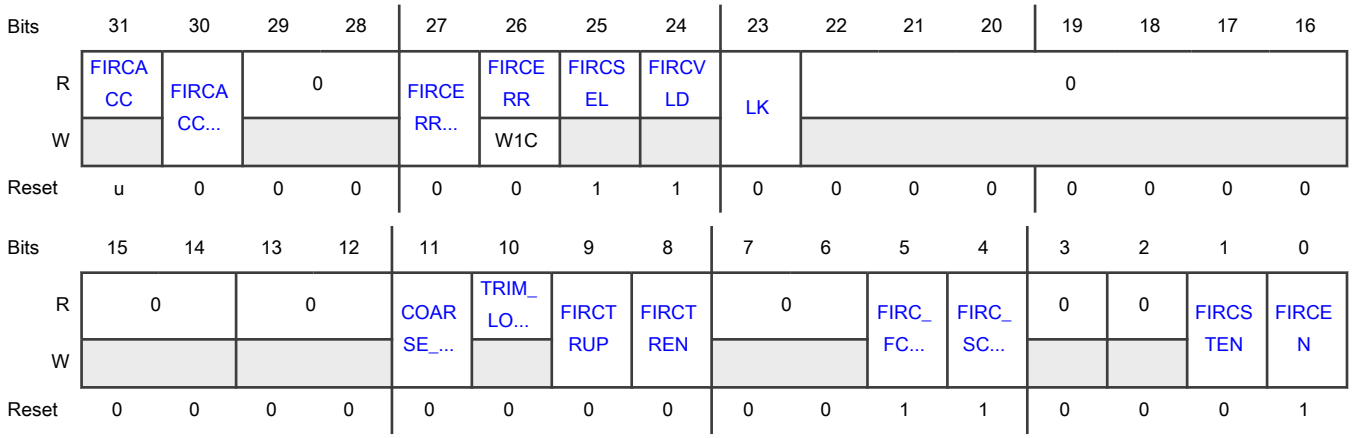
Offset

Register	Offset
FIRCCSR	300h

Function

Contains control and status fields for the FIRC clock source.

Diagram



Fields

Field	Function
31 FIRCACC	FIRC Frequency Accurate Indicates if the FIRC clock source is accurate. 0b - FIRC is not enabled or clock is not accurate. 1b - FIRC is enabled and output clock is accurate. The clock is accurate after 4096 clock cycles of 144 MHz (RANGE=1) or 1365 clock cycles of 48 MHz(RANGE=0) from the FIRC analog.
30 FIRCACC_IE	FIRC Accurate Interrupt Enable Generates an interrupt when FIRCACC is asserted. 0b - FIRCACC interrupt is not enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - FIRCACC interrupt is enabled
29-28 —	Reserved
27 FIRCERR_IE	FIRC Clock Error Interrupt Enable Generates an interrupt when FIRCERR is asserted. 0b - FIRCERR interrupt is not enabled 1b - FIRCERR interrupt is enabled
26 FIRCERR	FIRC Clock Error This flag is reset on Chip POR only. You can also clear this flag by writing 1. 0b - Error not detected with the FIRC trimming 1b - Error detected with the FIRC trimming
25 FIRCSEL	FIRC Selected Indicates if the FIRC clock source is selected as the system clock source. 0b - FIRC is not the system clock source 1b - FIRC is the system clock source
24 FIRCVLD	FIRC Valid status Indicates if the FIRC clock source is valid. 0b - FIRC is not enabled or clock is not valid. 1b - FIRC is enabled and output clock is valid. The clock is valid after there is an output clock from the FIRC analog.
23 LK	Lock Register You can clear or set this field at any time. Use this field to prevent runaway code from changing FIRC clock configurations. 0b - Control Status Register can be written 1b - Control Status Register cannot be written
22-14 —	Reserved
13-12 —	Reserved
11	Coarse Auto Trim Bypass Use this bit is to bypass the coarse trim step when auto-trimming.

Table continues on the next page...

Table continued from the previous page...

Field	Function
COARSE_TRIM_BYPASS	0b - FIRC coarse auto trim is not bypassed 1b - FIRC coarse auto trim is bypassed
10 TRIM_LOCK	FIRC TRIM LOCK When FIRCTREN and FIRCTRUP are enabled, the TRIM_LOCK register indicates when auto trimming is complete and output FIRC frequency has locked to target FIRC range. **NOTE** • This field is automatically cleared if FIRCTREN and FIRCTRUP are not set. • TRIM_LOCK is not set when FIRCTCFG[TRIMSRC] = 2'b00, i.e., using full speed USB as the trim source. 0b - FIRC auto trim not locked to target frequency range 1b - FIRC auto trim locked to target frequency range
9 FIRCTRUP	FIRC Trim Update Allows the FIRCSTAT to be updated by the auto-trimming hardware. 0b - Disables FIRC trimming updates 1b - Enables FIRC trimming updates
8 FIRCTREN	FIRC 144 MHz Trim Enable (FIRCCFG[RANGE]=1) Enables the auto tirm of FIRC 144 MHz by an external clock source. 0b - Disables trimming FIRC to an external clock source 1b - Enables trimming FIRC to an external clock source
7-6 —	Reserved
5 FIRC_FCLK_PERIPH_EN	FIRC 144 MHz Clock to peripherals Enable Enables the FIRC 144 MHz clock for peripheral use. 0b - FIRC 144 MHz to peripherals is disabled 1b - FIRC 144 MHz to peripherals is enabled
4 FIRC_SCLK_PERIPH_EN	FIRC 48 MHz Clock to peripherals Enable Enables the FIRC 48 MHz clock for peripheral use. 0b - FIRC 48 MHz to peripherals is disabled 1b - FIRC 48 MHz to peripherals is enabled
3 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 —	Reserved
1 FIRCSTEN	FIRC Stop Enable Enables the FIRC clock source in Deep Sleep mode if FIRCEN is set. 0b - FIRC is disabled in Deep Sleep mode 1b - FIRC is enabled in Deep Sleep mode
0 FIRCEN	FIRC Enable Enables the FIRC clock source. 0b - FIRC is disabled 1b - FIRC is enabled

34.7.1.14 FIRC Configuration Register (FIRCCFG)

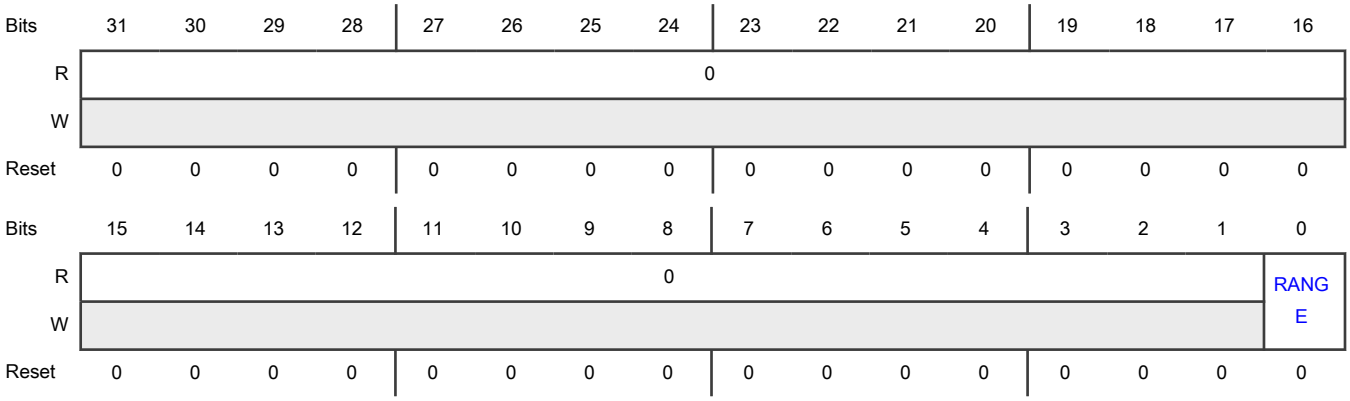
Offset

Register	Offset
FIRCCFG	308h

Function

Controls the clock frequency range select for the FIRC clock source.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 RANGE	Frequency Range Use this field to select the 48 MHz or 144 MHz FIRC clock when used as system clock. **NOTE** This field can be changed while FIRC clock is enabled. 0b - 48 MHz FIRC clock selected 1b - 144 MHz FIRC clock selected

34.7.1.15 FIRC Trim Configuration Register (FIRCTCFG)

Offset

Register	Offset
FIRCTCFG	30Ch

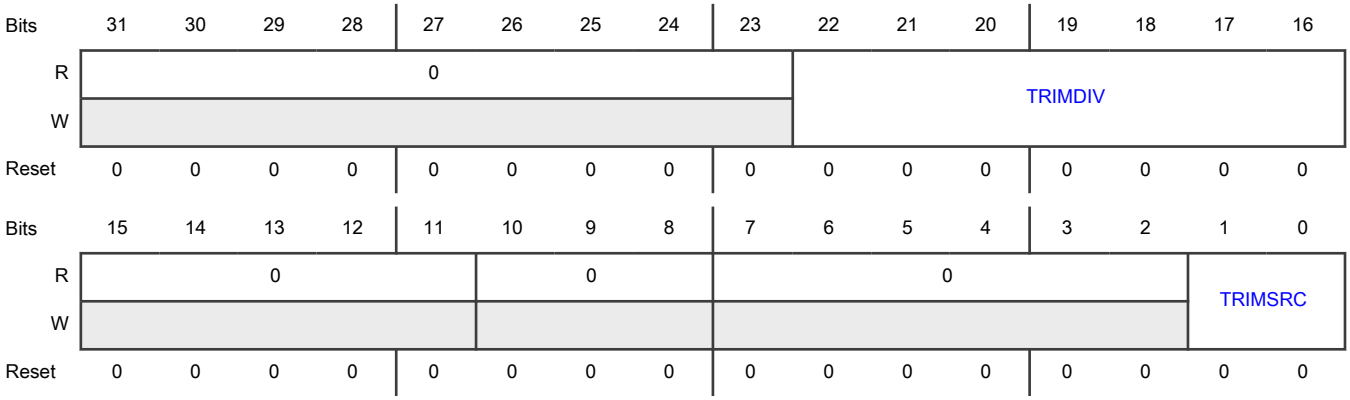
Function

Contains the auto trim clock source select and trim clock divider control for the FIRC clock source.

NOTE

You cannot change this register when FIRC tuning is enabled. When the FIRC tuning is enabled, writes to this register are ignored and there is no transfer error.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-16 TRIMDIV	FIRC Trim Predivider Divider of SOSC for FIRC trimming. **NOTE** When selecting SOSC as the FIRC trimming source, the TRIMDIV register must be set to correct div ratio to generate 1 MHz output reference trimming clock. **NOTE** TRIMDIV is an N-Divider supporting a div ratio of 1 (TRIMDIV=00h) to a div ratio of 128 (TRIMDIV=7Fh).
15-11 —	Reserved
10-8 —	Reserved
7-2 —	Reserved
1-0 TRIMSRC	Trim Source Configures the external clock source to tune the FIRC. This field must be configured before programming the FIRCSTAT register for trim update. 00b - USB0 Start of Frame (1 kHz). This option does not use TRIMDIV 01b - Reserved 10b - SOSC. This option requires that SOSC be divided using the TRIMDIV field to get a frequency of 1 MHz. 11b - ROSC. 32.768 kHz

34.7.1.16 FIRC Trim Register (FIRCTRIM)

Offset

Register	Offset
FIRCTRIM	310h

Function

The FIRC Trim Register holds the trim values for the FIRC clock source. This register is loaded from IFR during reset. These values are used for trimming the two range (RANGE=0 and RANGE=1) frequencies of FIRC.

NOTE

Writes to this register are protected by TRIM LOCK register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		TRIMSTART						0						TRIMTEMP	
W																
Reset	0	0	u[1]	u	u	u	u	u	0	0	0	0	0	0	u[1]	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		TRIMCOAR						TRIMFINE							
W																
Reset	0	0	u[1]	u	u	u	u	u	u[1]	u	u	u	u	u	u	u

1. Reset values are loaded out of IFR.

Fields

Field	Function
31-30 —	Reserved
29-24 TRIMSTART	Trim Start Current DAC adjustment of the replica cco current of start-up circuit.
23-18 —	Reserved
17-16 TRIMTEMP	Trim Temperature Temperature coefficient compensation.
15-14 —	Reserved
13-8 TRIMCOAR	Trim Coarse TRIMCOAR bits are used to coarsely trim the FIRC Clock to within approximately ±3.2% of the target frequency.
7-0 TRIMFINE	Trim Fine Current DAC adjustment of the CCO current. TRIMFINE bits are used to fine trim the FIRC Clock to within approximately ±0.25% of the target frequency.

34.7.1.17 FIRC Auto-trimming Status Register (FIRCSTAT)

Offset

Register	Offset
FIRCSTAT	318h

Function

This register is loaded from IFR during reset.

Trim values are uploaded to this register by FIRC auto-trimming. FIRC auto-trimming is enabled when FIRCTREN=1 and FIRCTRUP=1. When FIRCTREN=1 and FIRCTRUP=0 (write 10 or 11 to TRIMSRC), writes to this register are allowed and values written to this register are used to trim FIRC clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		TRIMCOAR						TRIMFINE							
W																
Reset	0	0	u[1]	u	u	u	u	u	u[1]	u	u	u	u	u	u	u

1. Reset values are loaded out of IFR.

Fields

Field	Function
31-16 —	Reserved
15-14 —	Reserved
13-8 TRIMCOAR	Trim Coarse Use this field for the current DAC adjustment of the base current and to coarsely trim the FIRC Clock to within approximately ±3.2% of the target 144 MHz frequency.
7-0 TRIMFINE	Trim Fine Use this field for the current DAC adjustment of the CCO current and to trim the FIRC Clock to within approximately ±0.25% of the target 144 MHz frequency.

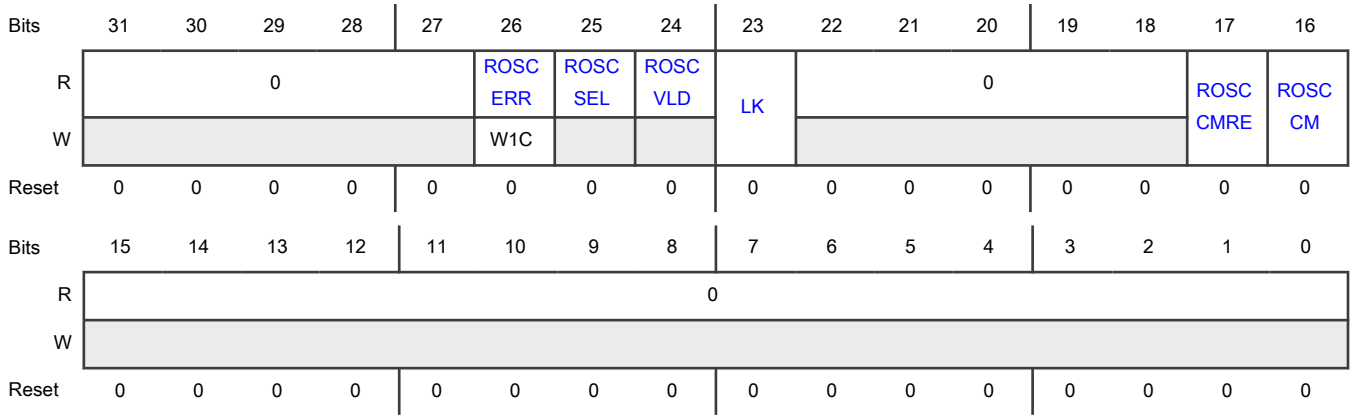
34.7.1.18 ROSC Control Status Register (ROSCCSR)

Offset

Register	Offset
ROSCCSR	400h

Function
Contains control and status bits for the ROSC clock source.

Diagram



Fields

Field	Function
31-27 —	Reserved
26 ROSCERR	ROSC Clock Error This flag is reset on Chip POR only. You can also clear this flag by writing 1. 0b - ROSC Clock Monitor is disabled or has not detected an error 1b - ROSC Clock Monitor is enabled and detected an RTC loss of clock error
25 ROSCSEL	ROSC Selected Indicates if the ROSC clock source is selected as the system clock source. 0b - ROSC is not the system clock source 1b - ROSC is the system clock source
24 ROSCVLD	ROSC Valid Indicates if the ROSC clock source is valid. 0b - ROSC is not enabled or clock is not valid

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - ROSC is enabled and output clock is valid
23 LK	Lock Register Locks this register so that it cannot be written to. **NOTE** This field can be cleared or set at any time. 0b - Control Status Register can be written 1b - Control Status Register cannot be written
22-18 —	Reserved
17 ROSCCMRE	ROSC Clock Monitor Reset Enable Enables the ROSCERR to generate a reset. 0b - Clock monitor generates an interrupt when an error is detected 1b - Clock monitor generates a reset when an error is detected
16 ROSCCM	ROSC Clock Monitor Enables the clock monitor when ROSCVLD is set. When the clock monitor is disabled in a low power mode, it remains disabled until the clock valid flag is set following exit from the low power mode. **NOTE** SIRC is the reference clock used to monitor the ROSC. You must program the SIRC clock to be enabled in order to monitor the ROSC. SIRC is automatically disabled in Power Down and Deep Power Down modes, and the clock monitor of the ROSC is disabled. 0b - ROSC clock monitor is disabled 1b - ROSC clock monitor is enabled
15-0 —	Reserved

34.7.1.19 APLL Control Status Register (APLLCSR)

Offset

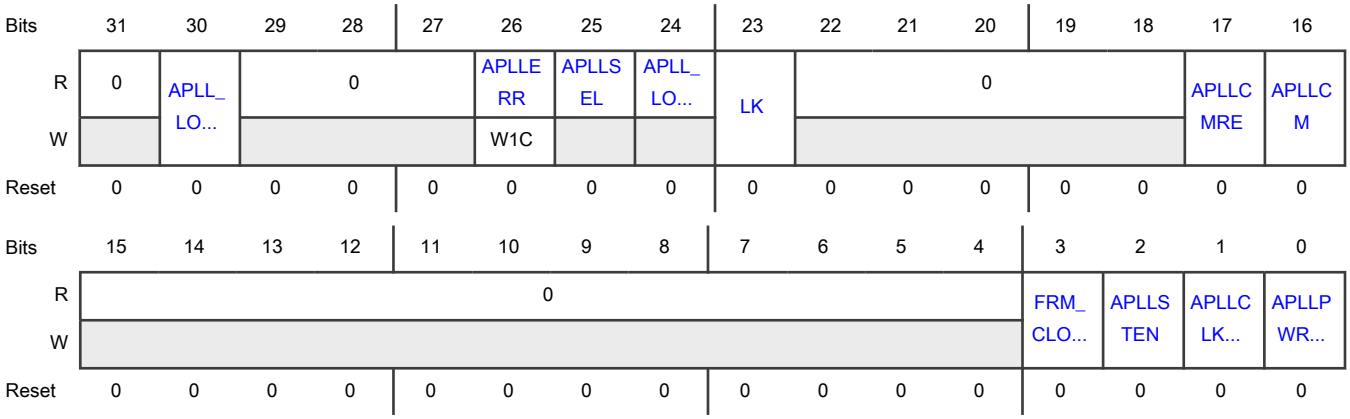
Register	Offset
APLLCSR	500h

Function
Contains control and status fields for the APLL clock source.

NOTE

After the PLL is powered on, do not change the configuration registers except for fields that can be modified on-the-fly.

Diagram



Fields

Field	Function
31 —	Reserved
30 APLL_LOCK_IE	APLL LOCK Interrupt Enable Generates an interrupt when APLL_LOCK is asserted. 0b - APLL_LOCK interrupt is not enabled 1b - APLL_LOCK interrupt is enabled
29-27 —	Reserved
26 APLLERR	APLL Clock Error This flag is reset on-chip POR only. You can also clear this flag by writing 1. **NOTE** This error is only set when FRMDET flag is generated in the free running mode. 0b - APLL Clock Monitor is disabled or has not detected an error 1b - APLL Clock Monitor is enabled and detected an error
25 APLLSEL	APLL Selected Indicates if the APLL clock source is selected as the system clock source. 0b - APLL is not the system clock source 1b - APLL is the system clock source

Table continues on the next page...

Table continued from the previous page...

Field	Function
24 APLL_LOCK	APLL LOCK Indicates when the APLL clock is locked. This flag is set when the number of reference clocks reaches LOCK_TIME after power on. APLL_LOCK clears with any of these conditions: • APLL power down • SOSC error when selected as the reference clock • Writes to APLLCTRL, APLLNDIV, APLLMDIV, APLLSSCG0, or APLLSSCG1 0b - APLL is not powered on or not locked 1b - APLL is locked
23 LK	Lock Register Locks this register so that it cannot be written to. **NOTE** You can clear or set this field at any time. 0b - Control Status Register can be written 1b - Control Status Register cannot be written
22-18 —	Reserved
17 APLLCMRE	APLL Clock Monitor Reset Enable Enables the APLLERR generate reset. **NOTE** This field is only used to monitor whether the FRMDET flag is generated in free running mode. 0b - Clock monitor generates an interrupt when an error is detected 1b - Clock monitor generates a reset when an error is detected
16 APLLCM	APLL Clock Monitor Enables the clock monitor. If the clock source is disabled in a low power mode, then the clock monitor is also disabled in that mode. The clock monitor is always disabled in Power Down and Deep Power Down modes. When the clock monitor is disabled in a low power mode, it remains disabled until the PLL_LOCK is set following exit from the low power mode. **NOTE** This field is only used to monitor whether the FRMDET flag is generated in free running mode. 0b - APLL Clock Monitor is disabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - APLL Clock Monitor is enabled
15-4 —	Reserved
3 FRM_CLOCKSTABLE	Free running mode clock stable **NOTE** Use this for free running mode when F_{in} / N = F_{ref} < 100 kHz. Only write 1 to this field after the PLL output frequency is stable. 0b - Free running mode clockstable is disabled 1b - Free running mode clockstable is enabled
2 APLLSTEN	APLL Stop Enable Enables the APLL clock source in Deep Sleep mode if APLLCLKEN and APLLPWREN are set. 0b - APLL is disabled in Deep Sleep mode 1b - APLL is enabled in Deep Sleep mode
1 APLLCLKEN	APLL Clock Enable Enables the APLL clock source. Use this field to enable the analog PLL to send out the clock. You can configure APLLCLKEN and APLLPWREN at the same time because hardware only enables the clock when PLL_LOCK=1. **NOTE** Ensure APLLCLKEN is enabled before using PLL clock. 0b - APLL clock is disabled 1b - APLL clock is enabled
0 APLLPWREN	APLL Power Enable Powers up the APLL clock source. 0b - APLL clock is powered off 1b - APLL clock is powered on

34.7.1.20 APLL Control Register (APLLCTRL)

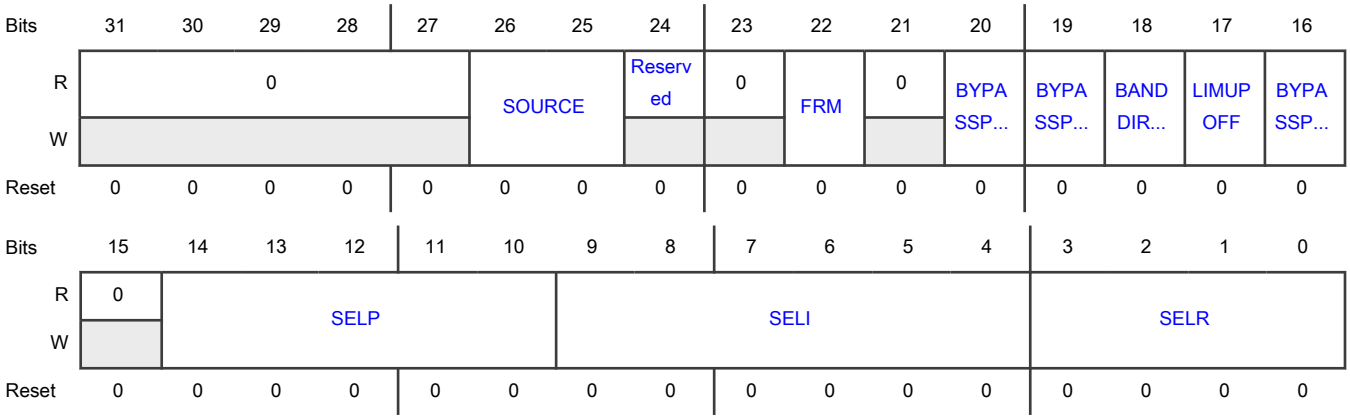
Offset

Register	Offset
APLLCTRL	504h

Function

Contains control fields for the analog PLL.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-25 SOURCE	Clock Source Configures the input clock source (clkin) for the APLL. **NOTE** A good quality input clock is necessary for good PLL performance. The input clock on clkin, optionally divided by the predivider ratio N, gives the reference frequency F_{ref} for the PLL-loop ($F_{ref} = F_{in} / N$). For optimal performance, use a reference frequency within the frequency range 5 MHz to 50 MHz. 00b - SOSC 01b - FIRC 48 MHz clock. FIRC_SCLK_PERIPH_EN must be set to use FIRC 48 MHz clock. 10b - Reserved 11b - No clock
24 —	Reserved
23 —	Reserved
22 FRM	Free Running Mode Enable In FRM mode, PLL monitors the input clock clkin. If the input clock disappears for more than six periods, the CCO of the PLL starts running free. The output frequency is within +1% and -20% for at least 200 µs. After 200 µs of self-clocking, the frequency keeps going down.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If the PLL is in free running mode, FRMDET is high. **NOTE** The free running mode only operates for reference frequencies above 100 kHz (F_{in} / N = F_{ref} > 100 kHz). To use the free running mode for F_{in} / N = F_{ref} < 100 kHz, only make FRM_CLOCKSTABLE = 1 after the PLL output frequency is stable. The free running mode is designed so that if the input clock occurs again, the PLL goes out of free running and is locked. Because the output clock has drifted away, overshoot and lock time have to be taken into account. **NOTE** When FRM=1, and PLL has detected a loss of input clock for more than six periods, the APLLERR flag is set. 0b - Free running mode disabled 1b - Free running mode enabled
21 —	Reserved
20 BYPASSPOSTDIV	Bypass of the postdivider Enables the bypass of the postdivider. **NOTE** Bypass the postdivider also bypasses the divide-by-2 divider in the postdivider. 0b - Use the postdivider. 1b - Bypass of the postdivider
19 BYPASSPREDIV	Bypass of the predivider Enables the bypass of the predivider. 0b - Use the predivider. 1b - Bypass of the predivider.
18 BANDDIRECT	Control of the bandwidth of the PLL. Enables the direct control of the bandwidth. 0b - The bandwidth is changed synchronously with the feedback-divider 1b - Modifies the bandwidth of the PLL directly
17 LIMUPOFF	Up Limiter In spread spectrum and fractional applications, LIMUPOFF must be set to 1. In other applications, LIMUPOFF = 0. 0b - Application set to non-Spectrum and Fractional applications.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Application set to Spectrum and Fractional applications.
16 BYPASSPOSTDIV2	Bypass of Divide-by-2 Divider Bypass of the divide-by-2 divider in the postdivider. 0b - Use the divide-by-2 divider in the postdivider 1b - Bypass of the divide-by-2 divider in the postdivider
15 —	Reserved
14-10 SELP	Bandwidth select P (proportional) value. See related content in Selecting bandwidth and PLL bandwidth settings.
9-4 SELI	Bandwidth select I (integration) value. See related content in Selecting bandwidth and PLL bandwidth settings.
3-0 SELR	Bandwidth select R (resistor) value. See related content in Selecting bandwidth and PLL bandwidth settings.

34.7.1.21 APLL Status Register (APLLSTAT)

Offset

Register	Offset
APLLSTAT	508h

Function

Contains status fields for the analog PLL.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											FRMDET	PDIVACK	MDIVACK	NDIVACK	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-5 —	Reserved
4 FRMDET	Free running detector (active high) In FRM mode, PLL monitors the input clock clkin. If the input clock disappears for more than six periods, the CCO of the PLL starts running free. The output frequency is within +1% and -20% for at least 200 µs. After 200 µs of self-clocking, the frequency keeps going down. If the PLL is in free running mode, FRMDET is high. 0b - Free running is not detected 1b - Free running is detected
3 PDIVACK	Postdivider(P) ratio change acknowledge. PDIVACK is response to the PREQ that the postdivider (P) ratio change is accepted by the analog PLL. 0b - The postdivider (P) ratio change is not accepted by the analog PLL 1b - The postdivider (P) ratio change is accepted by the analog PLL
2 MDIVACK	Feedback(M) divider ratio change acknowledge. MDIVACK is response to the MREQ that the feedback (M) ratio change is accepted by the analog PLL. 0b - The feedback (M) ratio change is not accepted by the analog PLL 1b - The feedback (M) ratio change is accepted by the analog PLL
1 NDIVACK	Predivider(N) ratio change acknowledge. NDIVACK is response to the NREQ that the predivider (N) ratio change is accepted by the analog PLL. 0b - The predivider (N) ratio change is not accepted by the analog PLL 1b - The predivider (N) ratio change is accepted by the analog PLL
0 —	Reserved

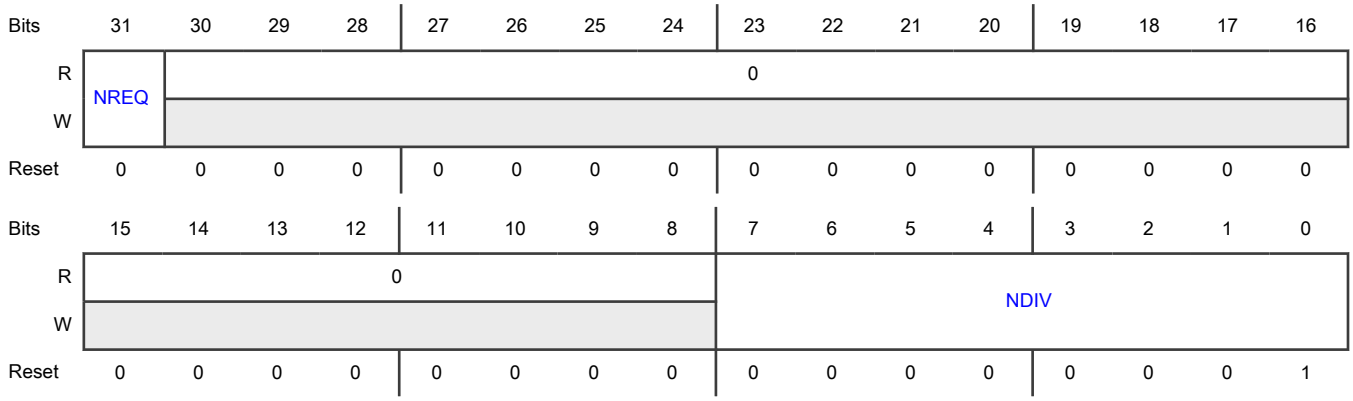
34.7.1.22 APLL N Divider Register (APLLNDIV)

Offset

Register	Offset
APLLNDIV	50Ch

Function
Contains predivider ratio and change request fields for on-the-fly update.

Diagram



Fields

Field	Function
31 NREQ	Predivider ratio change request. Normally, you must program the divider ratio of the different dividers (MDIV, NDIV, PDIV) when PLL is in reset state (APLLPWREN = 0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol. Normally, you must program the divider ratio for MDIV, NDIV, and PDIV when the PLL is in a reset state (APLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to NDIV to select a new divider ratio 2. Write 1 to NREQ to request the ratio change 3. APLLSTAT[NDIVACK] returns 1 to confirm the ratio change is accepted 4. Write 0 to NREQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. APLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for APLL_LOCK to set again to ensure the PLL is locked. 0b - Predivider ratio change is not requested 1b - Predivider ratio change is requested
30-8 —	Reserved
7-0 NDIV	Predivider divider ratio (N-divider). N-divider supports a divider ratio of 1 (NDIV=01h) to a divider ratio of 255 (NDIV=FFh). **NOTE** Do not write 0 to this field.

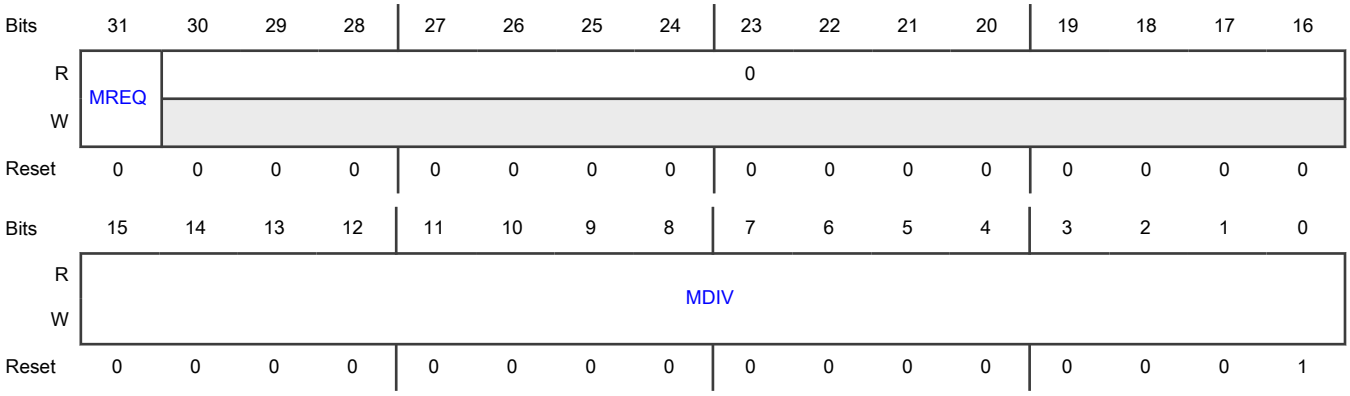
34.7.1.23 APLL M Divider Register (APLLMDIV)

Offset

Register	Offset
APLLMDIV	510h

Function
Contains feedback divider ratio and change request fields for on-the-fly update.

Diagram



Fields

Field	Function
31 MREQ	Feedback ratio change request. Normally, you must program the divider ratio for MDIV, NDIV, and PDIV when the PLL is in a reset state (APLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to MDIV to select a new divider ratio 2. Write 1 to MREQ to request the ratio change 3. APLLSTAT[MDIVACK] returns 1 to confirm the ratio change is accepted 4. Write 0 to MREQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. APLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for APLL_LOCK to set again to ensure the PLL is locked.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Feedback ratio change is not requested 1b - Feedback ratio change is requested
30-16 —	Reserved
15-0 MDIV	Feedback divider divider ratio (M-divider). M-divider supports a divider ratio of 1 (MDIV=0001h) to a divider ratio of 65535 (MDIV=FFFFh). **NOTE** Do not write 0 to this field.

34.7.1.24 APLL P Divider Register (APLLPDIV)

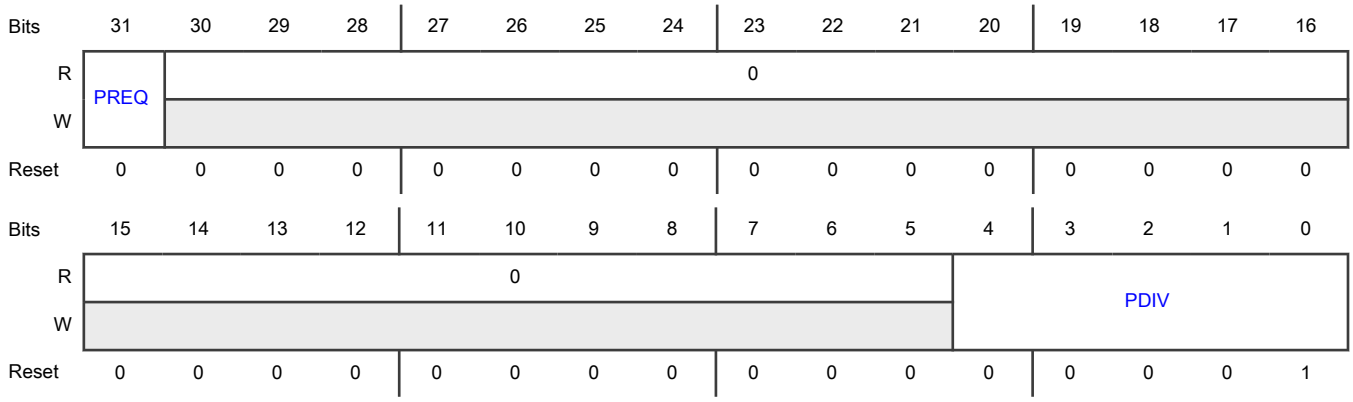
Offset

Register	Offset
APLLPDIV	514h

Function

Contains post divider ratio and change request fields for on-the-fly update.

Diagram



Fields

Field	Function
31 PREQ	Postdivider ratio change request

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Normally, you must program the divider ratio for MDIV, NDIV, and PDIV when the PLL is in a reset state (APLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to PDIV to select a new divider ratio 2. Write 1 to PREQ to request the ratio change 3. APLLSTAT[PDIVACK] returns 1 to confirm the ratio change is accepted 4. Write 0 to PREQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. APLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for APLL_LOCK to set again to ensure the PLL is locked. 0b - Postdivider ratio change is not requested 1b - Postdivider ratio change is requested
30-5 —	Reserved
4-0 PDIV	Postdivider divider ratio (P-divider) P-divider supports a divider ratio of PDIV*2, where PDIV is from 1 (PDIV=1h) to 31 (PDIV=1Fh). BYPASSPOSTDIV2 can be set to bypass the divide-by-2 in the postdivider. **NOTE** Do not write 0 to this field.

34.7.1.25 APLL LOCK Configuration Register (APLLLOCK_CNFG)

Offset

Register	Offset
APLLLOCK_CNFG	518h

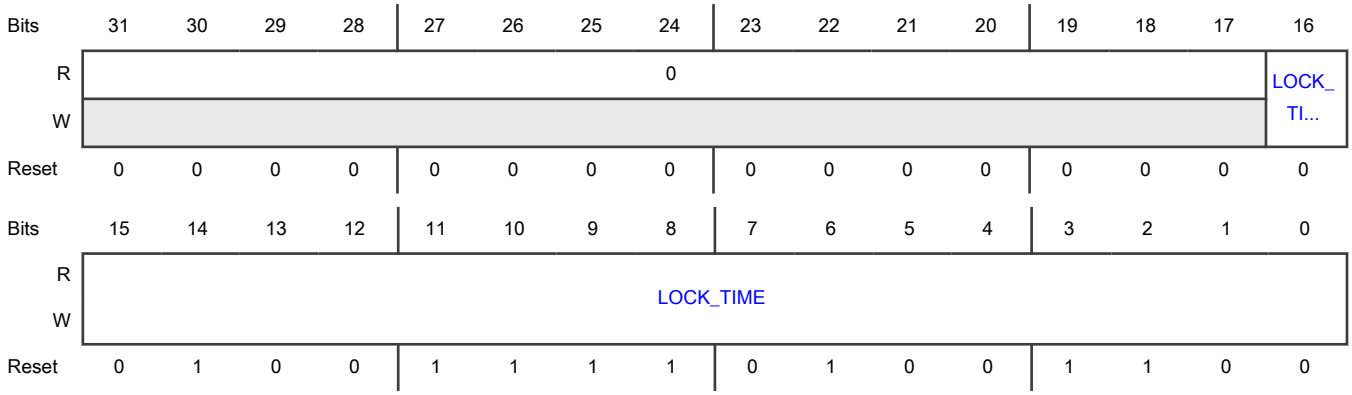
Function

Contains the number of reference clocks to count before APLL is considered locked and valid.

NOTE

Writes to this register are protected by the TRIM_LOCK register register.

Diagram



Fields

Field	Function
31-17 —	Reserved
16-0 LOCK_TIME	Configures the number of reference clocks to count before APLL is considered locked. **NOTE** The lock time programmed in this register must be equal to the PLL 500 µs lock time plus the 300 refclk count startup: LOCK_TIME = 500 µs/T_{ref} + 300, F_{ref} = F_{in}/N (input frequency divided by predivider ratio). **NOTE** Every time F_{ref} changes, you must program a new LOCK_TIME value.

34.7.1.26 APLL SSCG Status Register (APLLSSCGSTAT)

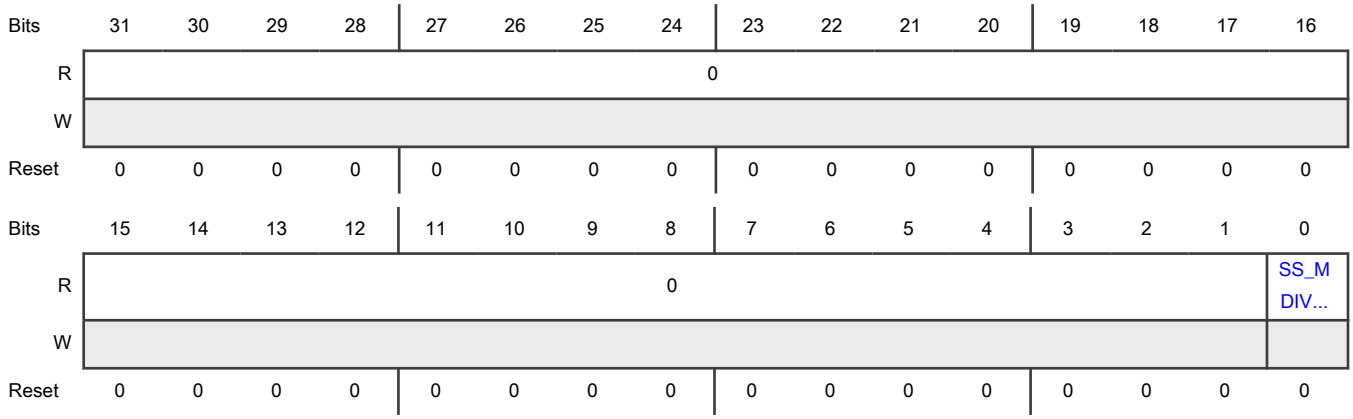
Offset

Register	Offset
APLLSSCGSTAT	520h

Function

Contains status field for the SSCG block.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 SS_MDIV_ACK	SS_MDIV change acknowledge SS_MDIV_ACK responses to the SS_MDIV_REQ. Indicates that the SS_MDIV, MF, MR, and MC change is accepted by the analog PLL. 0b - The SS_MDIV, MF, MR, and MC ratio change is not accepted by the analog PLL 1b - The SS_MDIV, MF, MR, and MC ratio change is accepted by the analog PLL

34.7.1.27 APLL Spread Spectrum Control 0 Register (APLLSSCG0)

Offset

Register	Offset
APLLSSCG0	524h

Function

Contains SS_MDIV.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SS_MDIV_LSB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SS_MDIV_LSB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SS_MDIV_LSB	SS_MDIV Represents the average feedback divider ratio $M_{average}$. SS_MDIV[32:25] is the integer part of the feedback divider ratio. SS_MDIV[24:0] is the fractional part of the feedback divider ratio. Average feedback-divider ratio $M_{average}$: $M_{average} = \text{SS_MDIV}[32:25]_{dec} + 2^{-25} \times SS_MDIV[24:0]_{dec} + DITHER \times 2^{-26} = 2^{-25} \times SS_MDIV_{dec} + DITHER \times 2^{-26}$ stepsize = $1/(2^{25} * M_{average})$

34.7.1.28 APLL Spread Spectrum Control 1 Register (APLLSSCG1)

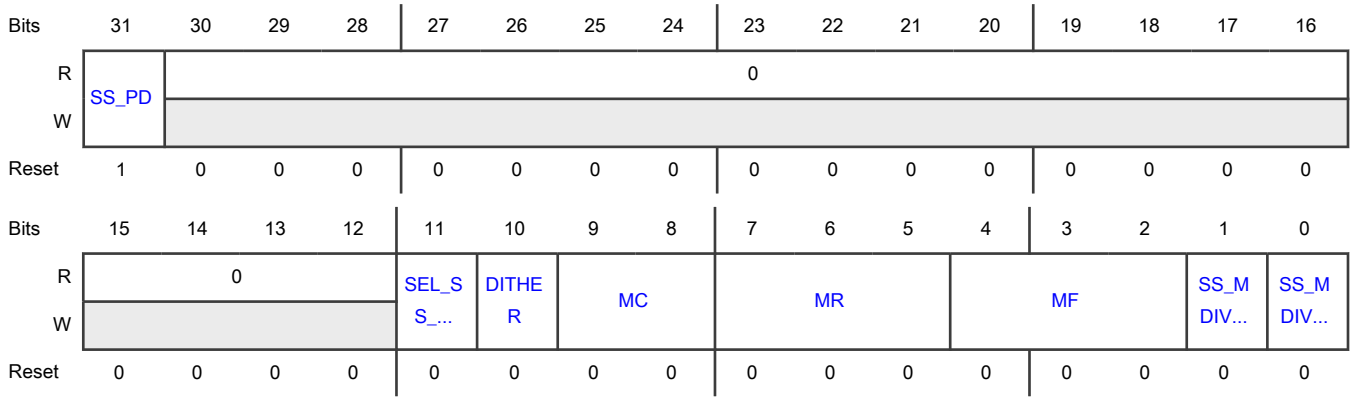
Offset

Register	Offset
APLLSSCG1	528h

Function

Contains SS_MDIV[32] and other control fields for the SSCG.

Diagram



Fields

Field	Function
31 SS_PD	SSCG Power Down Puts SSCG into power down. **NOTE** You must configure DITHER, MC, MF, MR, and SS_MDIV[32:0] before SS_PD is cleared. 0b - SSCG is powered on 1b - SSCG is powered off
30-12 —	Reserved
11 SEL_SS_MDIV	SS_MDIV select. Selects the SS_MDIV value. 0b - Feedback divider ratio is MDIV[15:0] 1b - Feedback divider ratio is SS_MDIV[32:0]
10 DITHER	Dither Enable Enables dithering between two modulation frequencies in a random way. This decreases the probability that the modulated waveform occurs with the same phase on a particular point on the screen. 0b - Dither is not enabled 1b - Dither is enabled
9-8 MC	Modulation Waveform Control Compensation for the low-pass filtering of the PLL in order to get a triangular modulation at the output of the PLL and a flat frequency spectrum. **NOTE** The recommended setting for the modulation waveform control is MC[1:0] = 10.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - MC[1:0] no compensation 11b - MC[1:0] maximum compensation
7-5 MR	Modulation Depth Control Controls the programmable frequency modulation depth according to the following equation. Average feedback-divider ratio: $$fmod_{pk\text{-}pk} = \frac{F_{ref} \times k_{ss}}{F_{sso}} = \frac{k_{ss}}{SS_MDIV[32:25]_{dec}}$$ • MR[2:0] = 000 - k_{SS} = 0 (no spread spectrum) • MR[2:0] = 001 - k_{SS} ≈ 0.5 • MR[2:0] = 010 - k_{SS} ≈ 0.75 • MR[2:0] = 011 - k_{SS} ≈ 1 • MR[2:0] = 100 - k_{SS} ≈ 1.5 • MR[2:0] = 101 - k_{SS} ≈ 2 • MR[2:0] = 110 - k_{SS} ≈ 3 • MR[2:0] = 111 - k_{SS} ≈ 4
4-2 MF	Modulation Frequency Control Controls the programmable modulation frequency. The below modulation frequencies are examples based on a 4 MHz reference frequency (F_{ref}). fm = F_{ref} /N_{SS} • MF[2:0] = 000 - N_{SS} = 512 (fm ≈ 7.8 kHz) • MF[2:0] = 001 - N_{SS} ≈ 384 (fm ≈ 10.4 kHz) • MF[2:0] = 010 - N_{SS} = 256 (fm ≈ 15.6 kHz) • MF[2:0] = 011 - N_{SS} = 128 (fm ≈ 31.3 kHz) • MF[2:0] = 100 - N_{SS} = 64 (fm ≈ 62.5 kHz) • MF[2:0] = 101 - N_{SS} = 32 (fm ≈ 125 kHz) • MF[2:0] = 110 - N_{SS} ≈ 24 (fm ≈ 166.6 kHz) • MF[2:0] = 111 - N_{SS} = 16 (fm ≈ 250 kHz)
1 SS_MDIV_REQ	SS_MDIV[32:0] change request. Change request for SS_MDIV, MF, MR, and MC. Normally, you must program the divider ratio for SS_MDIV, MF, MR, and MC when the PLL is in a reset state (APLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to SS_MDIV, MF, MR, and MC to select a new divider ratio

Table continues on the next page...

Table continued from the previous page...

Field	Function
	2. Write 1 to SS_MDIV_REQ to request the ratio change 3. SS_MDIV_ACK returns 1 to confirm the ratio change is accepted 4. Write 0 to SS_MDIV_REQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. APLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for APLL_LOCK to set again to ensure the PLL is locked. 0b - SS_MDIV change is not requested 1b - SS_MDIV change is requested
0 SS_MDIV_MSB	SS_MDIV[32] See APLLSSCG0[SS_MDIV_LSB].

34.7.1.29 APLL Override Register (APLL_OVRD)

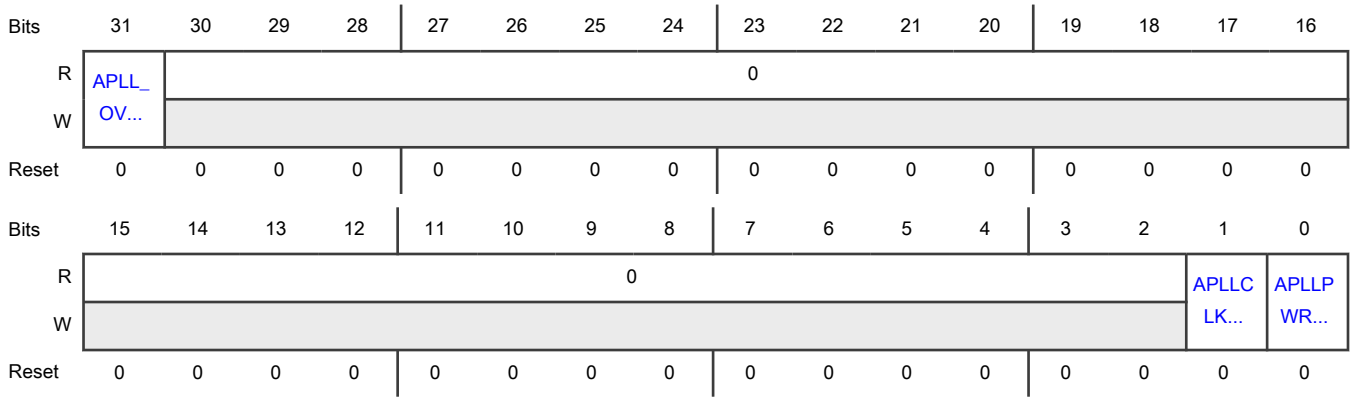
Offset

Register	Offset
APLL_OVRD	5F4h

Function

Contains override fields for the analog PLL.

Diagram



Fields

Field	Function
31 APLL_OVRD_EN	APLL Override Enable Enables the override control for APLL 0b - APLL override is disabled 1b - APLL override is enabled
30-2 —	Reserved
1 APLLCLKEN_OVRD	APLL Clock Enable Override if APLL_OVRD_EN=1 Enables the APLL clock source. 0b - APLL clock is disabled 1b - APLL clock is enabled
0 APLLPWREN_OVRD	APLL Power Enable Override if APLL_OVRD_EN=1 Powers up the APLL clock source. NOTE Write 0 to SS_PD to use the Spread Spectrum function. 0b - APLL clock is powered off 1b - APLL clock is powered on

34.7.1.30 SPLL Control Status Register (SPLLCSR)

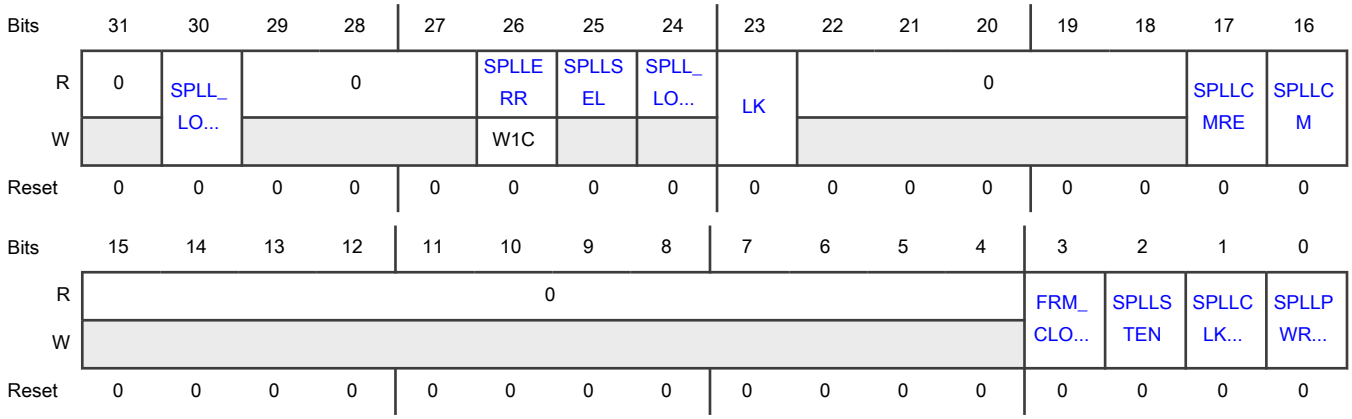
Offset

Register	Offset
SPLLCSR	600h

Function
Contains control and status fields for SPLL clock source.

NOTE
After PLL is powered on, do not change the configuration registers except for fields that can be modified on-the-fly.

Diagram



Fields

Field	Function
31 —	Reserved
30 SPLL_LOCK_IE	SPLL LOCK Interrupt Enable Generates an interrupt when SPLL_LOCK is asserted. 0b - SPLL_LOCK interrupt is not enabled 1b - SPLL_LOCK interrupt is enabled
29-27 —	Reserved
26 SPLLERR	SPLL Clock Error This flag is reset on Chip POR only. You can also clear this flag by writing 1. **NOTE** This field is only set when FRMDET flag is generated in free running mode. 0b - SPLL Clock Monitor is disabled or has not detected an error 1b - SPLL Clock Monitor is enabled and detected an error
25 SPLLSEL	SPLL Selected This flag indicates if the SPLL clock source is selected as the system clock source. 0b - SPLL is not the system clock source 1b - SPLL is the system clock source
24 SPLL_LOCK	SPLL LOCK Indicates when the SPLL clock is locked. This flag is set when the number of reference clocks reaches LOCK_TIME after power on.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	SPLL_LOCK clears with any of these conditions: • SPLL power down • SOSC error when selected as the reference clock • Writes to SPLLCTRL, SPLLNDIV, SPLLMDIV, SPLLSSCG0, or SPLLSSCG1 **NOTE** Use the SPLL_LOCK field to verify that the SPLL is locked after power-on. This field is cleared when you modify the SPLLCTRL, SPLLNDIV, SPLLMDIV, SPLLPDIV, SPLLSSCG0, or SPLLSSCG1 configurations on-the-fly. In this case, you must wait for SPLL_LOCK to set again to ensure PLL is locked before using the PLL clock. **NOTE** To monitor the SPLL clock, use the SPLLCM field to ensure that the SPLL Clock Monitor is enabled. 0b - SPLL is not powered on or not locked 1b - SPLL is locked
23 LK	Lock Register Locks this register so that it cannot be written to. **NOTE** You can clear or set this field at any time. 0b - Control Status Register can be written 1b - Control Status Register cannot be written
22-18 —	Reserved
17 SPLLCMRE	SPLL Clock Monitor Reset Enable Enables the SPLLERR generate reset. **NOTE** This field is only used to monitor whether the FRMDET flag is generated in free running mode. 0b - Clock monitor generates an interrupt when an error is detected 1b - Clock monitor generates a reset when an error is detected
16 SPLLCM	SPLL Clock Monitor Enables the clock monitor. If the clock source is disabled in a low power mode, then the clock monitor is also disabled in that mode. The clock monitor is always disabled in Power Down and Deep Power Down modes. When the clock monitor is disabled in a low power mode, it remains disabled until the PLL_LOCK is set following exit from that mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field is only used to monitor whether the FRMDET flag is generated in free running mode. 0b - SPLL Clock Monitor is disabled 1b - SPLL Clock Monitor is enabled
15-4 —	Reserved
3 FRM_CLOCKSTABLE	Free running mode clock stable **NOTE** Use this field for free running mode when F_{in} / N = F_{ref} < 100 kHz. Only write 1 to this field after the PLL output frequency is stable. 0b - Free running mode clockstable is disabled 1b - Free running mode clockstable is enabled
2 SPLLSTEN	SPLL Stop Enable Enables the SPLL clock source in Deep Sleep mode if SPLLCLKEN and SPLLPWREN are set. 0b - SPLL is disabled in Deep Sleep mode 1b - SPLL is enabled in Deep Sleep mode
1 SPLLCLKEN	SPLL Clock Enable Enables the SPLL clock source. Use this field to enable the analog PLL to send out the clock. You can configure SPLLCLKEN and SPLLPWREN at the same time because hardware only enables the clock when PLL_LOCK=1. **NOTE** Ensure SPLLCLKEN is enabled before using PLL clock. 0b - SPLL clock is disabled 1b - SPLL clock is enabled
0 SPLLPWREN	SPLL Power Enable Powers up the SPLL clock source. **NOTE** Write 0 to SS_PD to use the spread spectrum and fractional functions. 0b - SPLL clock is powered off 1b - SPLL clock is powered on

34.7.1.31 SPLL Control Register (SPLLCTRL)

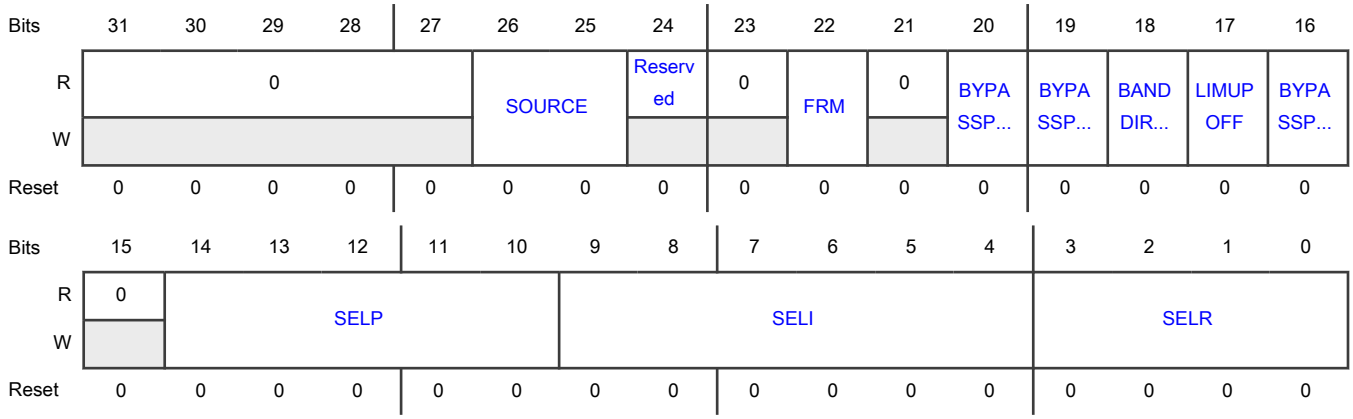
Offset

Register	Offset
SPLLCTRL	604h

Function

Contains control fields for the analog PLL.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-25 SOURCE	Clock Source Configures the input clock source (clkin) for the SPLL. **NOTE** A good-quality input clock is necessary for good PLL performance. The input clock on clkin, optionally divided by the predivider ratio N, gives the reference frequency F_{ref} for the PLL-loop ($F_{ref} = F_{in} / N$). For optimal performance, use a reference frequency within the frequency range 5 MHz to 50 MHz. 00b - SOSC 01b - FIRC 48 MHz clock. FIRC_SCLK_PERIPH_EN must be set to use FIRC 48 MHz clock. 10b - Reserved 11b - No clock
24	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
23 —	Reserved
22 FRM	Free Running Mode Enable In FRM mode, the PLL monitors the input clock clkin. If the input clock disappears for more than six periods, the CCO of the PLL starts running free. The output frequency is within +1% and -20% for at least 200 µs. After 200 µs of self-clocking, the frequency keeps going down. If the PLL is in free running mode, FRMDET is high. **NOTE** The free running mode only operates for reference frequencies above 100 kHz (F_{in} / N = F_{ref} > 100 kHz). To use the free running mode for F_{in} / N = F_{ref} < 100 kHz, only make FRM_CLOCKSTABLE = 1 after the PLL output frequency is stable. The free running mode is designed so that if the input clock occurs again, the PLL goes out of free running and is locked. Because the output clock has drifted away, overshoot and lock time have to be taken into account. **NOTE** When FRM=1, and PLL has detected a loss of input clock for more than six periods, SPLLERR flag is set. 0b - Free running mode disabled 1b - Free running mode enabled
21 —	Reserved
20 BYPASSPOSTDIV	Bypass of the postdivider. Enables the bypass of the postdivider. 0b - Use the postdivider 1b - Bypass of the postdivider
19 BYPASSPREDIV	Bypass of the predivider. Enables the bypass of the predivider. 0b - Use the predivider 1b - Bypass of the predivider
18 BANDDIRECT	Control of the bandwidth of the PLL. Enables the direct control of the bandwidth. 0b - The bandwidth is changed synchronously with the feedback-divider

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Modifies the bandwidth of the PLL directly
17 LIMUPOFF	Up Limiter. In spread spectrum and fractional applications, you must set LIMUPOFF to 1. In other applications, LIMUPOFF = 0. 0b - Application set to non-Spectrum and Fractional applications. 1b - Application set to Spectrum and Fractional applications.
16 BYPASSPOSTDIV2	Bypass of Divide-by-2 Divider Bypass of the divide-by-2 divider in the postdivider. 0b - Use the divide-by-2 divider in the postdivider. 1b - Bypass of the divide-by-2 divider in the postdivider
15 —	Reserved
14-10 SELP	Bandwidth select P (proportional) value. See related content in Selecting bandwidth and PLL bandwidth settings.
9-4 SELI	Bandwidth select I (integration) value. See related content in Selecting bandwidth and PLL bandwidth settings.
3-0 SELR	Bandwidth select R (resistor) value. See related content in Selecting bandwidth and PLL bandwidth settings.

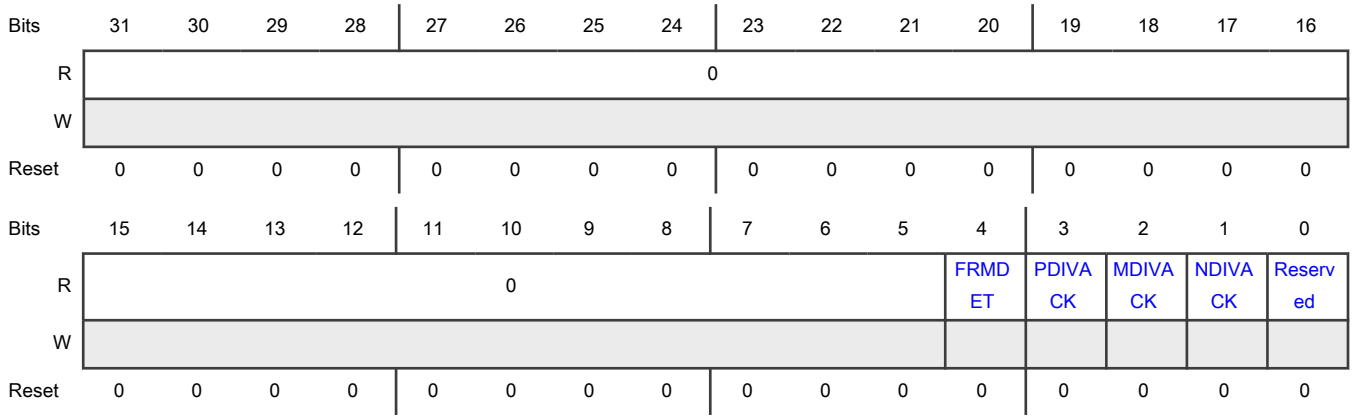
34.7.1.32 SPLL Status Register (SPLLSTAT)

Offset

Register	Offset
SPLLSTAT	608h

Function
Contains status fields for the analog PLL.

Diagram



Fields

Field	Function
31-5 —	Reserved
4 FRMDET	Free running detector (active high) In FRM mode, the PLL monitors the input clock clkin. If the input clock disappears for more than six periods, the CCO of the PLL starts running free. The output frequency is within +1% and -20% for at least 200 µs. After 200 µs of self-clocking, the frequency keeps going down. If the PLL is in free running mode, FRMDET is high. 0b - Free running is not detected 1b - Free running is detected
3 PDIVACK	Postdivider (P) ratio change acknowledge PDIVACK is response to the PREQ that the postdivider (P) ratio change is accepted by the analog PLL. 0b - The postdivider (P) ratio change is not accepted by the analog PLL 1b - The postdivider (P) ratio change is accepted by the analog PLL
2 MDIVACK	Feedback (M) divider ratio change acknowledge Indicates to the MREQ that the feedback (M) ratio change is accepted by the analog PLL. 0b - The feedback (M) ratio change is not accepted by the analog PLL. 1b - The feedback (M) ratio change is accepted by the analog PLL.
1 NDIVACK	Predivider (N) ratio change acknowledge Indicates to the NREQ that the predivider (N) ratio change is accepted by the analog PLL. 0b - The predivider (N) ratio change is not accepted by the analog PLL. 1b - The predivider (N) ratio change is accepted by the analog PLL.
0 —	Reserved

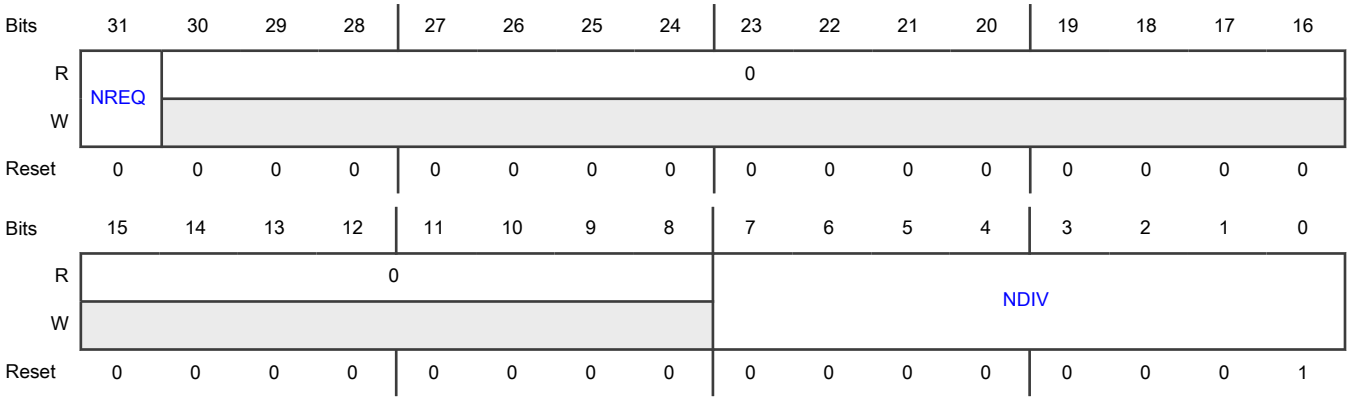
34.7.1.33 SPLL N Divider Register (SPLLNDIV)

Offset

Register	Offset
SPLLNDIV	60Ch

Function
Contains predivider ratio and change request fields for on-the-fly update.

Diagram



Fields

Field	Function
31 NREQ	Predivider ratio change request. Normally, you must program the divider ratio of the different dividers (MDIV, NDIV, PDIV) when PLL is in reset state (SPLLPWREN = 0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol. Normally, you must program the divider ratio for MDIV, NDIV, and PDIV when the PLL is in a reset state (SPLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to NDIV to select a new divider ratio 2. Write 1 to NREQ to request the ratio change 3. SPLLSTAT[NDIVACK] returns 1 to confirm the ratio change is accepted 4. Write 0 to NREQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. SPLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for SPLL_LOCK to set again to ensure the PLL is locked.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Predivider ratio change is not requested 1b - Predivider ratio change is requested
30-8 —	Reserved
7-0 NDIV	Predivider divider ratio (N-divider). N-divider supports a divider ratio of 1 (NDIV=01h) to a divider ratio of 255 (NDIV=FFh). **NOTE** Do not write 0 to this field.

34.7.1.34 SPLL M Divider Register (SPLLMDIV)

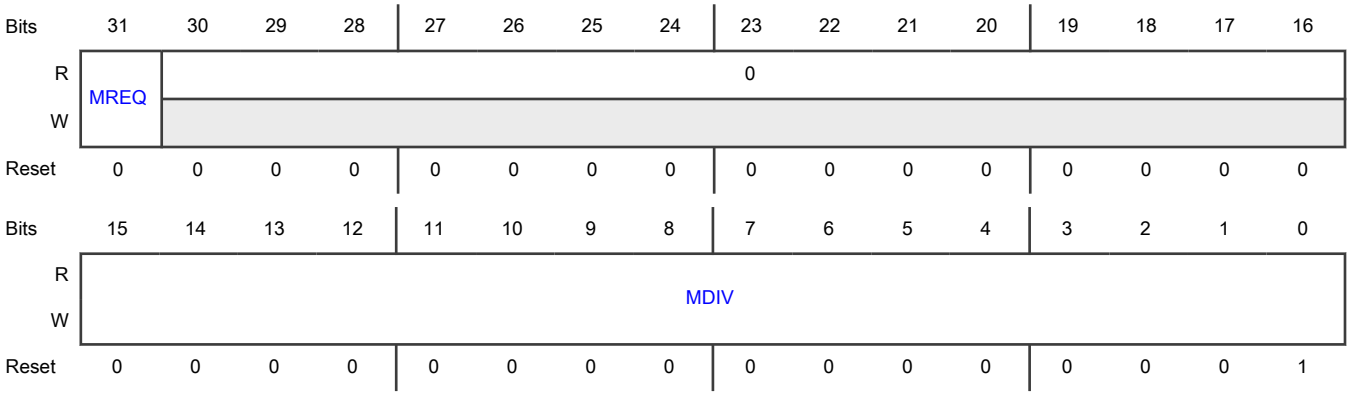
Offset

Register	Offset
SPLLMDIV	610h

Function

Contains feedback divider ratio and change request fields for on-the-fly update.

Diagram



Fields

Field	Function
31 MREQ	Feedback ratio change request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Normally, you must program the divider ratio for MDIV, NDIV, and PDIV when the PLL is in a reset state (SPLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to MDIV to select a new divider ratio 2. Write 1 to MREQ to request the ratio change 3. SPLLSTAT[MDIVACK] returns 1 to confirm the ratio change is accepted 4. Write 0 to MREQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. SPLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for SPLL_LOCK to set again to ensure the PLL is locked. 0b - Feedback ratio change is not requested 1b - Feedback ratio change is requested
30-16 —	Reserved
15-0 MDIV	Feedback divider divider ratio (M-divider). M-divider supports a divider ratio of 1 (MDIV=0001h) to a divider ratio of 65535 (MDIV=FFFFh). **NOTE** Do not write 0 to this field.

34.7.1.35 SPLL P Divider Register (SPLLPDIV)

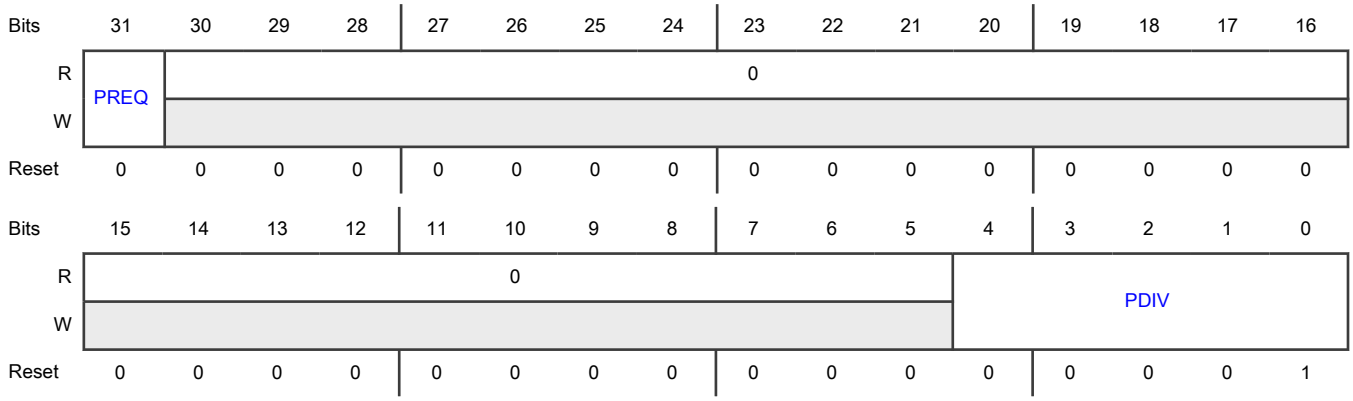
Offset

Register	Offset
SPLLPDIV	614h

Function

Contains postdivider ratio and change request fields for an on-the-fly update.

Diagram



Fields

Field	Function
31 PREQ	Postdivider ratio change request Normally, you must program the divider ratio for MDIV, NDIV, and PDIV when the PLL is in a reset state (SPLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to PDIV to select a new divider ratio 2. Write 1 to PREQ to request the ratio change 3. SPLLSTAT[PDIVACK] returns 1 to confirm the ratio change is accepted 4. Write 0 to PREQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. SPLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for SPLL_LOCK to set again to ensure the PLL is locked. 0b - Postdivider ratio change is not requested 1b - Postdivider ratio change is requested
30-5 —	Reserved
4-0 PDIV	Postdivider divider ratio (P-divider) P-divider supports a divider ratio of PDIV*2, where PDIV is from 1 (PDIV=1h) to 31 (PDIV=1Fh). BYPASSPOSTDIV2 can be set to bypass the divide by 2 in the postdivider. **NOTE** Do not write 0 to this field.

34.7.1.36 SPLL LOCK Configuration Register (SPLLLOCK_CNFG)

Offset

Register	Offset
SPLLLOCK_CNFG	618h

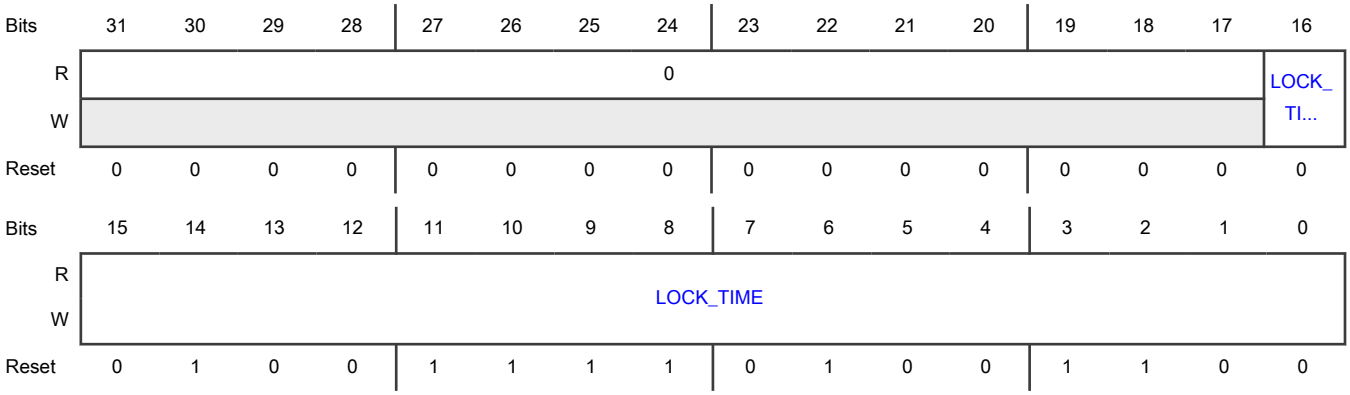
Function

Configures the number of reference clocks to count before SPLL is considered locked and valid.

NOTE

Writes to this register are protected by the TRIM_LOCK register.

Diagram



Fields

Field	Function
31-17 —	Reserved
16-0 LOCK_TIME	Configures the number of reference clocks to count before SPLL is considered locked. **NOTE** The lock time programmed in this register must be equal to meet the PLL 500 µs lock time plus the 300 refclk count startup, LOCK_TIME = 500 µs/T_{ref} + 300, F_{ref} = F_{in}/N (input frequency divided by predivider ratio). **NOTE** Every time F_{ref} changes, a new value LOCK_TIME must be programmed.

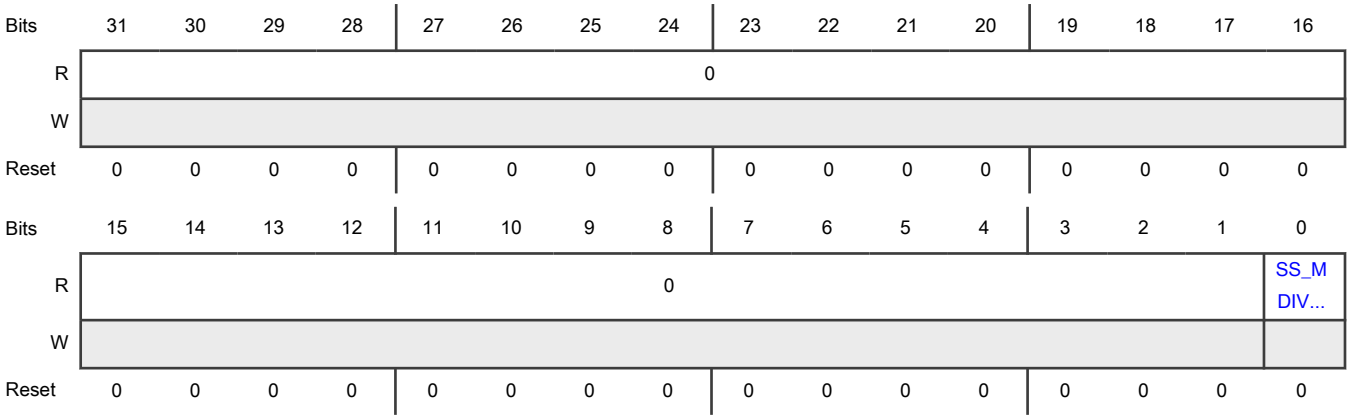
34.7.1.37 SPLL SSCG Status Register (SPLLSSCGSTAT)

Offset

Register	Offset
SPLLSSCGSTAT	620h

Function
Contains status field for the SSCG block.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 SS_MDIV_ACK	SS_MDIV change acknowledge Indicates to the SS_MDIV_REQ that the SS_MDIV, MF, MR, and MC change is accepted by the analog PLL. 0b - The SS_MDIV, MF, MR, and MC ratio change is not accepted by the analog PLL 1b - The SS_MDIV, MF, MR, and MC ratio change is accepted by the analog PLL

34.7.1.38 SPLL Spread Spectrum Control 0 Register (SPLLSSCG0)

Offset

Register	Offset
SPLLSSCG0	624h

Function
Contains SS_MDIV[31:0].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SS_MDIV_LSB															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SS_MDIV_LSB															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SS_MDIV_LSB	SS_MDIV[31:0] Represents the average feedback divider ratio $M_{average}$. SS_MDIV[32:25] is the integer part of the feedback divider ratio. SS_MDIV[24:0] is the fractional part of the feedback divider ratio. **NOTE** SS_MDIV[32:0] must not be programmed to value 0 for spread spectrum and fractional function. Average feedback-divider ratio $M_{average}$: $M_{average} = \text{SS_MDIV}[32:25]_{dec} + 2^{-25} \times SS_MDIV[24:0]_{dec} + DITHER \times 2^{-26} = 2^{-25} \times SS_MDIV_{dec} + DITHER \times 2^{-26}$ stepsize = $1/(2^{25} * M_{average})$

34.7.1.39 SPLL Spread Spectrum Control 1 Register (SPLLSSCG1)

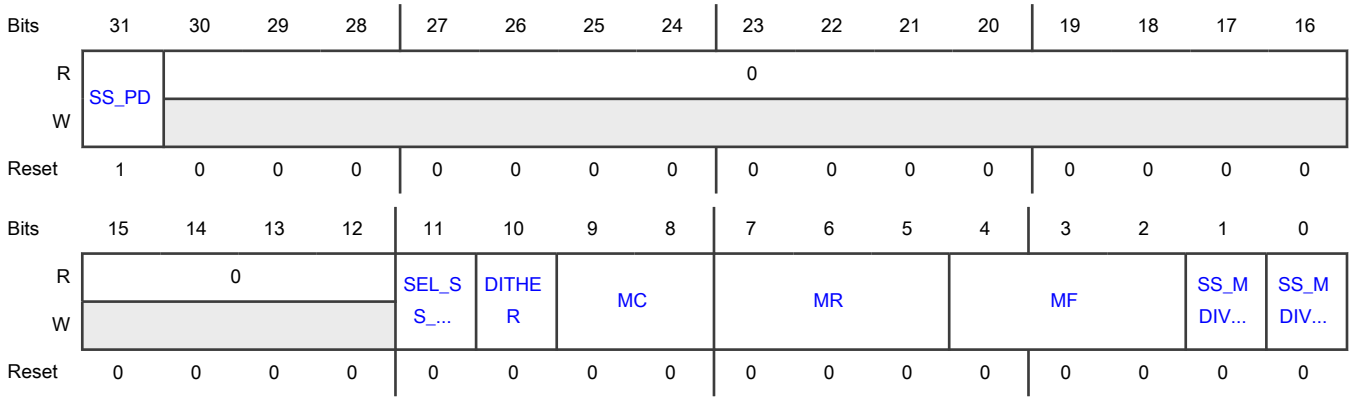
Offset

Register	Offset
SPLLSSCG1	628h

Function

Contains SS_MDIV[32] and other control fields for SSCG.

Diagram



Fields

Field	Function
31 SS_PD	SSCG Power Down Puts SSCG into power down. **NOTE** The configuration of DITHER, MC, MF, MR, and SS_MDIV[32:0] must be written before SS_PD is cleared. 0b - SSCG is powered on 1b - SSCG is powered off
30-12 —	Reserved
11 SEL_SS_MDIV	SS_MDIV select. Selects the SS_MDIV value. 0b - Feedback divider ratio is MDIV[15:0] 1b - Feedback divider ratio is SS_MDIV[32:0]
10 DITHER	Dither Enable Enables dithering between two modulation frequencies in a random way. This decreases the probability of the modulated waveform occurring with the same phase on a particular point on the screen. 0b - Dither is not enabled 1b - Dither is enabled
9-8 MC	Modulation Waveform Control Compensation for the low-pass filtering of the PLL in order to get a triangular modulation at the output of the PLL and a flat frequency spectrum. **NOTE** The recommended setting for the modulation waveform control is MC[1:0] = 10.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - MC[1:0] no compensation 11b - MC[1:0] maximum compensation
7-5 MR	Modulation Depth Control Controls the programmable frequency modulation depth according to the following equation. Average feedback-divider ratio: $$fmod_{pk\text{-}pk} = \frac{F_{ref} \times k_{ss}}{F_{sso}} = \frac{k_{ss}}{SS_MDIV[32:25]_{dec}}$$ • MR[2:0] = 000 - k_{SS} = 0 (no spread spectrum) • MR[2:0] = 001 - k_{SS} ≈ 0.5 • MR[2:0] = 010 - k_{SS} ≈ 0.75 • MR[2:0] = 011 - k_{SS} ≈ 1 • MR[2:0] = 100 - k_{SS} ≈ 1.5 • MR[2:0] = 101 - k_{SS} ≈ 2 • MR[2:0] = 110 - k_{SS} ≈ 3 • MR[2:0] = 111 - k_{SS} ≈ 4
4-2 MF	Modulation Frequency Control Controls the programmable modulation frequency. The below modulation frequencies are examples based on a 4 MHz reference frequency (F_{ref}). fm = F_{ref} /N_{SS} • MF[2:0] = 000 - N_{SS} = 512 (fm ≈ 7.8 kHz) • MF[2:0] = 001 - N_{SS} ≈ 384 (fm ≈ 10.4 kHz) • MF[2:0] = 010 - N_{SS} = 256 (fm ≈ 15.6 kHz) • MF[2:0] = 011 - N_{SS} = 128 (fm ≈ 31.3 kHz) • MF[2:0] = 100 - N_{SS} = 64 (fm ≈ 62.5 kHz) • MF[2:0] = 101 - N_{SS} = 32 (fm ≈ 125 kHz) • MF[2:0] = 110 - N_{SS} ≈ 24 (fm ≈ 166.6 kHz) • MF[2:0] = 111 - N_{SS} = 16 (fm ≈ 250 kHz)
1 SS_MDIV_REQ	SS_MDIV[32:0] change request. Change request for SS_MDIV, MF, MR, and MC. Normally, you must program the divider ratio for SS_MDIV, MF, MR, and MC when the PLL is in a reset state (SPLLWREN=0). However, you can also select the divider ratio on-the-fly with the help of a handshake protocol: 1. Write to SS_MDIV, MF, MR, and MC to select a new divider ratio

Table continues on the next page...

Table continued from the previous page...

Field	Function
	2. Write 1 to SS_MDIV_REQ to request the ratio change 3. SS_MDIV_ACK returns 1 to confirm the ratio change is accepted 4. Write 0 to SS_MDIV_REQ to end the ratio change request **NOTE** Use the reset method if more than one divider ratio must be changed. **NOTE** During the on-the-fly divider ratio update process, the PLL clock must be switched off from the system clock. SPLL_LOCK bit is cleared in this case. Every time F_{ref} changes, a new value LOCK_TIME must be programmed. Before using the PLL clock, you must wait for SPLL_LOCK to set again to ensure the PLL is locked. 0b - SS_MDIV change is not requested 1b - SS_MDIV change is requested
0 SS_MDIV_MSB	SS_MDIV[32] See SS_MDIV_LSB.

34.7.1.40 SPLL Override Register (SPLL_OVRD)

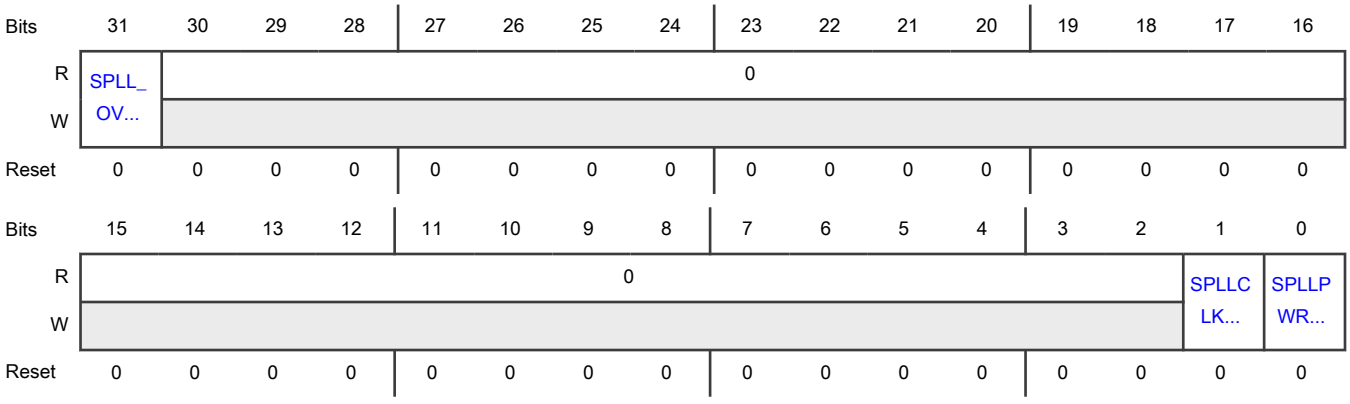
Offset

Register	Offset
SPLL_OVRD	6F4h

Function

Contains override fields for the analog PLL.

Diagram



Fields

Field	Function
31 SPLL_OVRD_EN	SPLL Override Enable Enable the override control for SPLL 0b - SPLL override is disabled 1b - SPLL override is enabled
30-2 —	Reserved
1 SPLLCLKEN_OVRD	SPLL Clock Enable Override if SPLL_OVRD_EN=1 Enable the SPLL clock source. 0b - SPLL clock is disabled 1b - SPLL clock is enabled
0 SPLLPWREN_OVRD	SPLL Power Enable Override if SPLL_OVRD_EN=1 Power up the SPLL clock source. **NOTE** Write 0 to SS_PD to use the Spread Spectrum function. 0b - SPLL clock is powered off 1b - SPLL clock is powered on

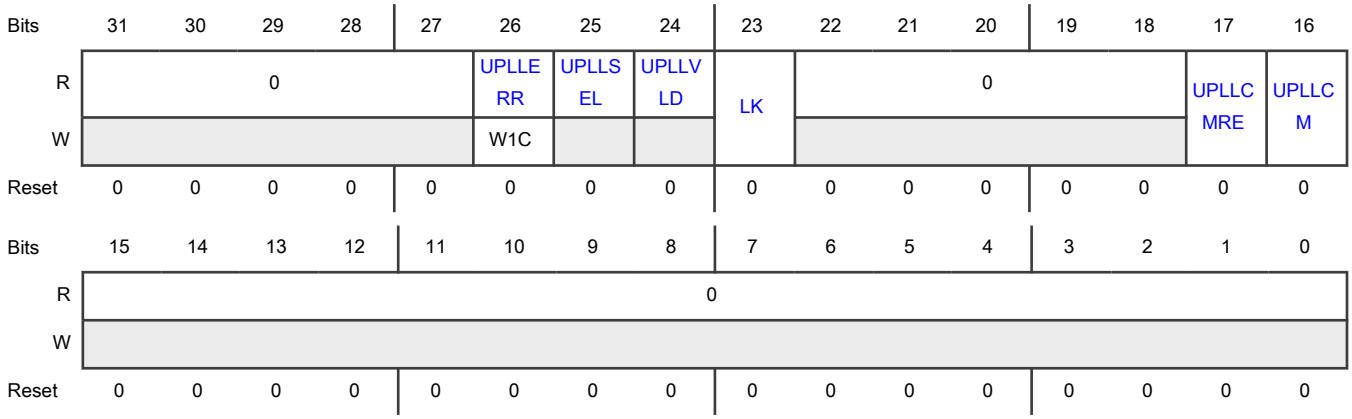
34.7.1.41 UPLL Control Status Register (UPLLCSR)

Offset

Register	Offset
UPLLCSR	700h

Function
Contains control and status bits for UPLL clock source.

Diagram



Fields

Field	Function
31-27 —	Reserved
26 UPLLERR	UPLL Clock Error This flag is reset on Chip POR only. 0b - UPLL Clock Monitor is disabled or has not detected an error 1b - UPLL Clock Monitor is enabled and detected an error
25 UPLLSEL	UPLL Selected Indicates whether the UPLL clock source is selected as the system clock source. 0b - UPLL is not the system clock source 1b - UPLL is the system clock source
24 UPLLVLD	UPLL Valid Indicates when the UPLL clock is valid. 0b - UPLL is not enabled or clock is not valid 1b - UPLL is enabled and output clock is valid
23 LK	Lock Register Locks this register so that it cannot be written to. **NOTE** You can clear or set this field at any time. 0b - Control Status Register can be written 1b - Control Status Register cannot be written
22-18	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
17 UPLLCMRE	UPLL Clock Monitor Reset Enable Enables the UPLLERR generate reset. 0b - Clock monitor generates an interrupt when an error is detected 1b - Clock monitor generates a reset when an error is detected
16 UPLLCM	UPLL Clock Monitor Enables the clock monitor. If the clock source is disabled in a low power mode, then the clock monitor is also disabled in that mode. The clock monitor is always disabled in Power Down and Deep Power Down modes. When the clock monitor is disabled in a low power mode, it remains disabled until the clock valid flag is set following exit from that mode. 0b - UPLL Clock Monitor is disabled 1b - UPLL Clock Monitor is enabled
15-0 —	Reserved

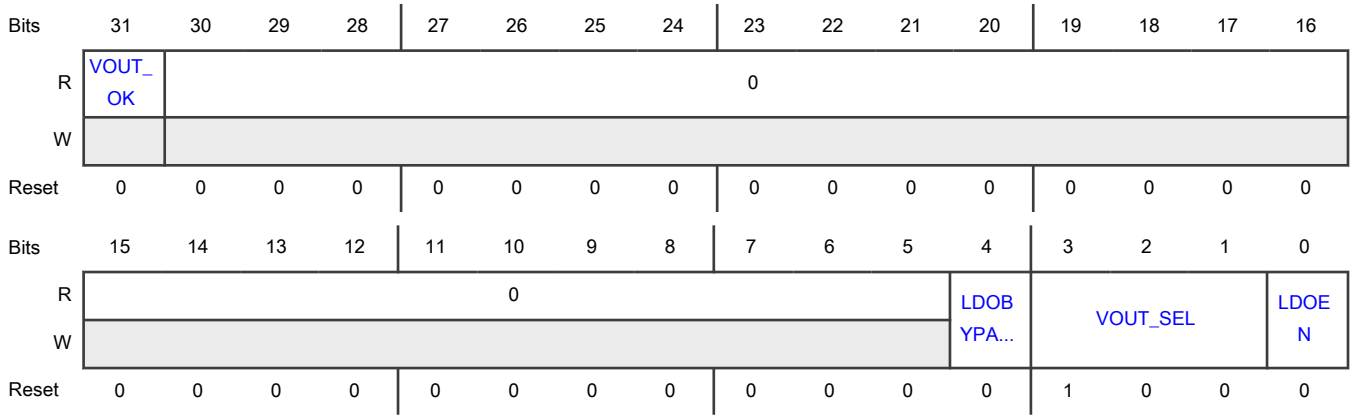
34.7.1.42 LDO Control and Status Register (LDOCSR)

Offset

Register	Offset
LDOCSR	800h

Function
Contains control bits and status bit for the LDO.

Diagram



Fields

Field	Function
31 VOUT_OK	LDO VOUT OK Inform. This flag shows the status of LDO output VOUT. 0b - LDO output VOUT is not OK 1b - LDO output VOUT is OK
30-5 —	Reserved
4 LDOBYPASS	LDO Bypass LDO is bypassed. **NOTE** Writes to this register are protected by the TRIM_LOCK register. 0b - LDO is not bypassed 1b - LDO is bypassed
3-1 VOUT_SEL	LDO output voltage select Sets the LDO output voltage level. **NOTE** Writes to this register are protected by the TRIM_LOCK register. 000b - VOUT = 1V 001b - VOUT = 1V 010b - VOUT = 1V 011b - VOUT = 1.05V 100b - VOUT = 1.1V 101b - VOUT = 1.15V 110b - VOUT = 1.2V 111b - VOUT = 1.25V
0 LDOEN	LDO Enable LDO is enabled. **NOTE** Disable LDO before entering the Power Down mode. 0b - LDO is disabled 1b - LDO is enabled

Chapter 35
Power Management

35.1 Introduction

This chapter describes the power modes that this chip supports. It also describes the power domains and voltage supply options connected to these power domains. For details about changing power modes and configuring those modes, see the CMC chapter.

35.2 Power domains

A power domain is a collection of circuits that can be powered off even when a voltage source is still supplied. This section shows the power domains within the chip and the placement of the modules within these domains.

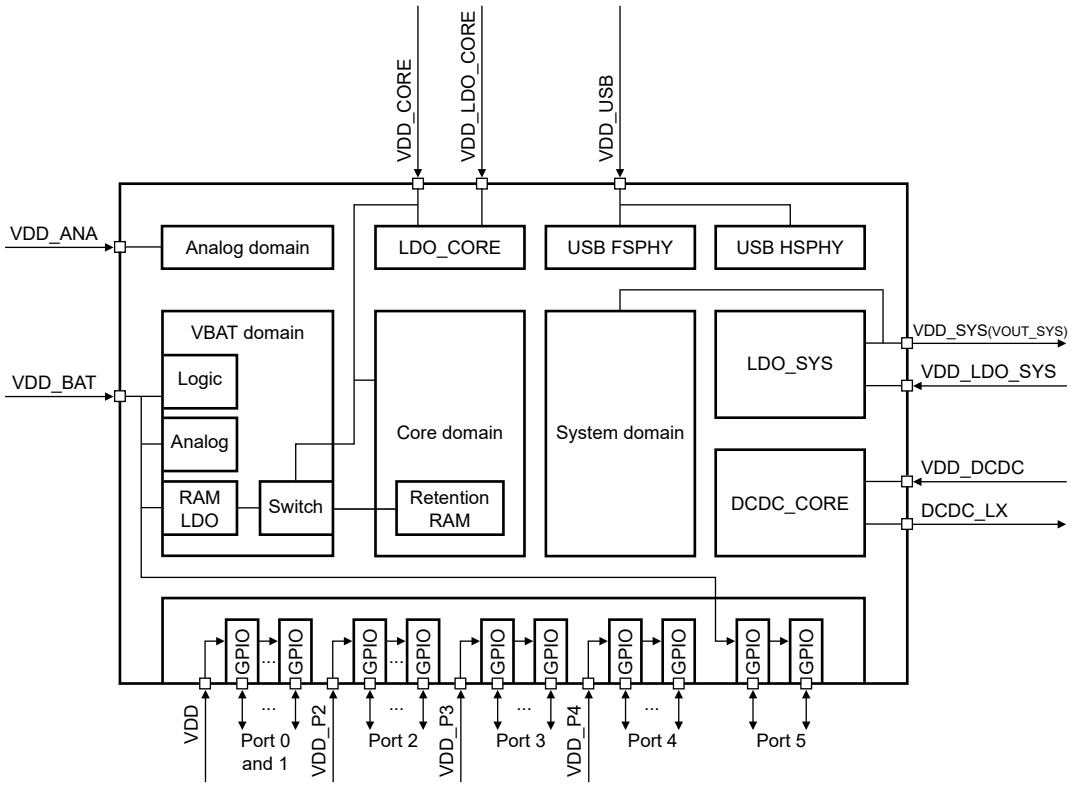


Figure 169. Power domains and voltage supplies

Table 333. Power domain assignments for modules

Power domain	Voltage supply	Module
USB	VDD_USB	USBFS PHY
		USBHS PHY
ANALOG	VDD_ANA, VSS_ANA	ADC0/1
		DAC0/1/2
		VREF0 (analog circuitry)

Table continues on the next page...

Table 333. Power domain assignments for modules (continued)

Power domain	Voltage supply	Module
		OPAMP0/1/2
CORE_MAIN	VDD_CORE	Cortex-M33 0/1
		CoolFlux BSP32
		NPU
		SmartDMA
		PowerQuad
		DSP
		FPU
		BSCAN
		DAP
		Debug Mailbox
		DWT
		ETB
		ETM
		ITM
		TPIU
		APB Bridge 0/1 and AIPS Bridge 0–4
		Multilayer AHB Matrix
		DMA0/1
		Peripheral input mux
		EVTG0
		Frequency measurement unit
		Mailbox
		MPU
		NVIC
		SEMA42
		SINC Filter
		SYSTICK
		uSDHC
		PLU
		CACHE-CODE
		FLASH (sense amps)

Table continues on the next page...

Table 333. Power domain assignments for modules (continued)

Power domain	Voltage supply	Module
		FMU0
		FlexSPI0
		FlexSPI cache
		FlexSPI cache performance monitor
		FlexSPI CACHE64 policy select
		LPCAC performance monitor
		NPX
		ROM-BOOT
		CRC0
		IPED with GCM
		GDET0/1
		MBC
		PKC
		Secure AHB control
		ELS
		TZM
		SCTIMER
		CTIMER1/2/3/4
		MRT0
		PWM0/1
		QDC0/1
		Subsecond counter (RTC_SUBSYSTEM)
		CAN0/1
		FlexIO0
		LP_FLEXCOMM2/3/4.../9
		I3C1
		SAI0/1
		USBDCD (digital circuitry)
		USBFS (digital circuitry)
		USBHS OTG
		ENET QoS
		EMVSIM

Table continues on the next page...

Table 333. Power domain assignments for modules (continued)

Power domain	Voltage supply	Module
		ADC0-GP (digital circuitry)
		DAC0/1-GP (digital circuitry)
		DAC2 (digital circuitry)
		OPAMP0/1/2 (digital circuitry)
		VREF0 (digital circuitry)
		GPIO1/2/3/4 (digital circuitry)
		PORT1/2/3/4 (digital circuitry)
		PINT0
CORE_SRAM	VDD_CORE	RAMX
		RAMA - RAMH
CORE_WAKE	VDD_CORE	WWDT0/1
		EWM0
		CTIMER0
		OSTIMER0
		UTICK0
		LP_FLEXCOMM0/1
		I3C0
		TSI0
		SWJ
		PORT0
		IOMUX0
		GPIO0
		CMP2
		SYSCON
		CMC
		Peripheral mux
		MICFIL
		SCG0
		FRO_12M
DCDC	VDD_DCDC	DCDC_CORE
VDD	VDD	Flash (Arrays)
		PORT0 (pins)

Table continues on the next page...

Table 333. Power domain assignments for modules (continued)

Power domain	Voltage supply	Module
		PORT1 (pins)
		CMP0/1-GP (analog circuitry)
		TSI (analog circuitry)
IO_2	VDD_P2	PORT2
IO_3	VDD_P3	PORT3
IO_4	VDD_P4	PORT4
LDO_CORE	VDD_LDO_CORE	LDO_CORE
SYSTEM	VDD_SYS	SPC0
		HVD/LVD
		IVS
		LDO_SYS
		WUU0
		CMP0/1
		LPTMR0/1
		PLL LDO (SOSC, PLL0, PLL1)
VBAT	VDD_BAT	32K OSC
		FRO_16K
		RAM LDO, optional supply for RAMA array
		Digital tamper
		VBAT wrapper
		RTC
		Wake-up timer (RTC_SUBSYSTEM)
		PORT 5 (8 pins)
		GPIO 5 (8 pins)

35.3 Power modes

The chip supports these modes:

- Active
- Sleep
- Deep Sleep
- Power Down
- Deep Power Down
- VBAT

Any reset returns the chip to Active mode.

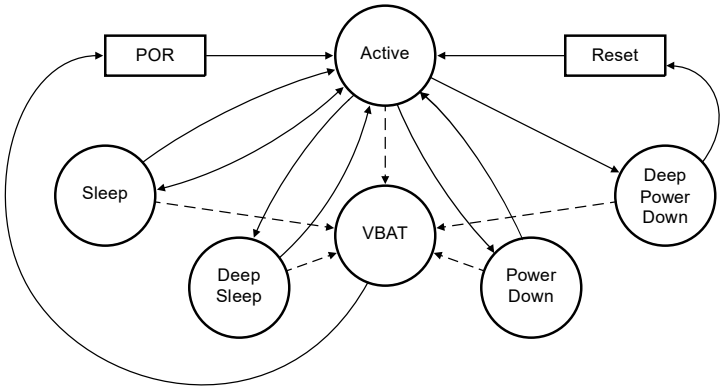


Figure 170. Power modes

35.3.1 Active mode

In Active mode, CPU execution is possible. There are three Active modes, OD (Over Drive), SD (Standard Drive), and MD (Mid Drive). In OD mode, VDD_CORE is 1.2 V; in SD mode, VDD_CORE is 1.1 V; and in MD mode, VDD_CORE is 1.0 V.

To achieve the performance requirements of a given system application while minimizing power consumption, Active mode permits the following power-saving options when possible:

- Configure the voltage level of the CORE subdomains to balance power and performance.
- Gate the clocks to unused modules.

35.3.2 Sleep mode

In Sleep mode, the CPU0 clock is off, and the system clock and bus clock remain on. Most modules can remain operational. To achieve the performance requirements of a given system application while minimizing power consumption, Sleep mode permits the following power-saving options when possible:

- Configure the voltage level of the CORE subdomains to balance power and performance.
- Individually configure SRAM arrays as Active or Deep Sleep via software.
- Gate the clocks to unused modules.

35.3.3 Deep Sleep mode

In Deep Sleep mode, CPU execution is halted. The core clock is gated off.

Deep Sleep mode supports the following behaviors based on different clock configurations:

- CPU clock, system clock, and bus clock are all off.
- Some modules can remain operational with low-power asynchronous clock sources and can serve as wake-up sources.
- System RAM is configured to State Retention mode.
- SRAM is in Deep Sleep mode.

An interrupt or wake-up event triggers waking from Deep Sleep mode.

Deep Sleep mode also supports a partial wake-up, where a wake-up event recovers a bus controller other than the CPU (for example, DMA). The chip automatically reenters Deep Sleep mode after this module finishes its task.

To achieve the performance requirements of a given system application while minimizing power consumption, Deep Sleep mode permits the following power-saving options when possible:

- Configure the voltage level of the CORE subdomains to balance power and performance.
- Gate the clocks to unused modules.

35.3.4 Power Down mode

Power Down mode places most of the chip into a static state. It is the lowest power mode that can retain all registers.

The chip wakes from Power Down mode through the interrupt routine or a wake-up event.

The core clock, system clock, and the bus clock are gated off.

In Power Down mode, the CORE_MAIN and CORE_WAKE subdomains are put into state retention mode.

External signals can wake the chip, via GPIO or VDD_SYS domain peripherals.

To balance different module clock frequencies and power leakage, the voltage level of CORE subdomains can be independently selected from the VDD_CORE voltage or the reduced voltage scaling (IVS) voltage.

Flash memory is powered off. Your software can individually configure SRAM arrays into deep sleep or shutdown.

35.3.5 Deep Power Down mode

The chip wakes from Deep Power Down mode through the Reset routine.

In Deep Power Down mode, the whole CORE domain, including all subdomains, is power-gated.

In Deep Power Down mode, the on-chip regulator for VDD_CORE is powered off. The VDD_SYS on-chip regulator is powered, but modules unrelated to the operation of RTC are clock-gated. To avoid extra leakage, the external power supply to any disabled on-chip regulator must be turned off.

External reset or VDD_SYS domain peripherals can wake the chip.

SRAMA can be retained in Deep Power Down mode.

35.3.6 VBAT mode

VBAT mode permits the lowest power level.

The POR sequence wakes the chip from VBAT mode.

Powering off VDD_SYS and VDD_CORE externally puts the chip into VBAT mode. For example, an external power switch can power off VDD_SYS and the regulator supplying VDD_CORE.

SRAMA can be retained in VBAT mode.

35.4 Module operation in low-power modes

The following tables show the functionality of each module in low-power modes.

Table 334. Overall operation in low-power modes

Module	Active	Sleep	Deep Sleep	Power Down	Deep Power Down	VBAT
CPU	On[1]	Static[2]	Static	Static	Off[3]	Off
CORE_MAIN domain	On	On	LP[4]	Static	Off	Off
CORE_WAKE domain	On	On	LP	LP/Static	Off	Off
System SRAM	On/Static[5]	On/Static	Static	Static/Off[6]	Off	Off
VBAT SRAM RAMA	On/Static	On/Static	Static	Static/Off	Static/Off	Static/Off
Flash	On/LP	On/LP	LP	Off	Off	Off

Table continues on the next page...

Table 334. Overall operation in low-power modes (continued)

Module	Active	Sleep	Deep Sleep	Power Down	Deep Power Down	VBAT
DCDC and LDO_CORE	On/Off	On/Off/LP	On/Off/LP	On/Off/LP	Off	Off
LDO and VDD_SYS	On	On/LP	On/LP	On/LP	On/LP	Off
Wake-up flow	N/A	Interrupt	Interrupt	Interrupt	Reset	Reset
Partial wake-up, no CPU	N/A	N/A	Supported (all sources)	Supported (optional wake domain or VDD_SYS and VBAT sources)	No	No
IVS	Off	Off	On/Off (SRAM only)	On/Off	Off	Off
CKMODE	N/A	0h/1h	Fh	Fh	Fh	N/A
LPMODE	N/A	0h	1h	3h	Fh	N/A

1. On means the module is functional and accessible via memory map. For analog module, it also means the module is enabled.
2. Static means the module is not active in state retention status.
3. Off means the module can be powered off. For analog module, it also means the module can be disabled.
4. LP means the module is in low power state (clock gated, asynchronous operation, etc). For digital module, it can be active with async functional clock.
5. CMC_SRAMDIS register controls configuration of SRAM arrays in Active and Sleep modes.
6. CMC_SRAMRET register controls configuration of SRAM arrays in Power Down mode.

Table 335. Module operation in low-power modes

Module	Sleep	Deep Sleep	Power Down[1]	Deep Power Down	VBAT
Core modules					
Cortex-M33 0	Static	Static	Static	Off	Off
Cortex-M33 1	On	Static	Static	Off	Off
Code-Cache	Static	Static	Static	Off	Off
SYSTICK	Static	Static	Static	Off	Off
NVIC	On	Static	Static	Off	Off
WIC	On	Static	Static	Off	Off
System modules					
Multilayer AHB matrix	On	Static	Static	Off	Off
SYSCON	On	Static	Static	Off	Off
PowerQuad	On	Static	Static	Off	Off
CoolFlux BSP32	On	Static	Static	Off	Off
NPU	On	Static	Static	Off	Off

Table continues on the next page...

Table 335. Module operation in low-power modes (continued)

Module	Sleep	Deep Sleep	Power Down[1]	Deep Power Down	VBAT
SmartDMA	On	Static	Static	Off	Off
DMA0, DMA1	On	LP/Static	LP/Static	Off	Off
WWDT0, WWDT1	On	LP/Static	LP/Static	Off	Off
CMC0	On	LP	LP/Static	Off	Off
SPC0	On	LP	LP	LP	Off
WUU0	On	LP	LP	LP	Off
PLU	On	LP/Static	Static	Off	Off
Clock modules					
FRO_144M	On/Off	On/Off	Off	Off	Off
FRO_12M	On/Off	On/Off	On/Off	Off	Off
FRO_16K	On/Off	On/Off	On/Off	On/Off	On/Off
OSC_RTC	On/Off	On/Off	On/Off	On/Off	On/Off
OSC_SYS	On/Off	On/Off	Off	Off	Off
PLL	On/Off	On/Off	Off	Off	Off
SCG	On	LP	LP/Static	Off	Off
Memory modules					
Flash array	On/LP	LP	Off	Off	Off
SRAM	On/Static	Static	Static/Off	Off	Off
RAMA (VBAT)	On/Static	Static	Static/Off	Static/Off	Static/Off
FlexSPI0	On	Static	Static	Off	Off
ROM	On	Static	Off	Off	Off
Security modules					
ELS	On	Static	Static	Off	Off
TDET	On	LP/Static	LP/Static	LP/Static	LP/Static
CRC0	On	Static	Static	Off	Off
PKC	On	Static	Static	Off	Off
GDET	On	Static	Static	Off	Off
Secure AHB control	On	Static	Static	Off	Off
CDOG0, CDOG1	Static	Static	Static	Off	Off
ITRC	On	Static	Static	Off	Off
Timer modules					
CTIMER0	On	LP/Static	LP/Static	Off	Off

Table continues on the next page...

Table 335. Module operation in low-power modes (continued)

Module	Sleep	Deep Sleep	Power Down[1]	Deep Power Down	VBAT
CTIMER1, CTIMER2, CTIMER3	On	LP/Static	Static	Off	Off
UTICK timer	On	LP/Static	LP/Static	Off	Off
OSTIMER	On	LP/Static	LP/Static	Off	Off
PWM0, PWM1	On	Static	Static	Off	Off
MRT	On	Static	Static	Off	Off
Subsecond counter (RTC_SUBSYSTEM)	On	LP/Static	Static	Off	Off
QDC0, QDC1	On	Static	Static	Off	Off
LPTMR0, LPTMR1	On	LP/Static	LP/Static	LP/Static	Off
RTC	On	LP/Static	LP/Static	LP/Static	LP/Static
Wake-up timer (RTC_SUBSYSTEM)	On	LP/Static	LP/Static	LP/Static	LP/Static
EWM0	On	Static	Static	Off	Off
FREQME	On	Static	Static	Off	Off
SCT	On	Static	Static	Off	Off
Communication modules					
USBFS0 Digital	On	LP/Static	Static	Off	Off
USBFS0 PHY	On/Off	On/Off	On/Off	On/Off	Off
LP_FLEXCOMM0, LP_FLEXCOMM1	On	LP/Static	LP/Static	Off	Off
LP_FLEXCOMM2–9	On	LP/Static	Static	Off	Off
SAI0, SAI1	On	LP/Static	Static	Off	Off
USBHS OTG Digital	On	Static	Static	Off	Off
USBHS PHY	On/Off	On/Off	Off	Off	Off
uSDHC	On	Static/LP	Static	Off	Off
ENET QoS	On	Static	Static	Off	Off
FlexIO0	On	LP/Static	Static	Off	Off
I3C0	On	LP/Static	LP/Static	Off	Off
I3C1	On	LP/Static	Static	Off	Off
CAN0, CAN1	On	LP/Static	Static	Off	Off

Table continues on the next page...

Table 335. Module operation in low-power modes (continued)

Module	Sleep	Deep Sleep	Power Down[1]	Deep Power Down	VBAT
EMVSIM	On	Static	Static	Off	Off
Human machine interface modules					
GPIO0	On	LP/Static	LP/Static	Off	Off
GPIO1–4	On	LP/Static	Static	Off	Off
GPIO5	On	LP/Static	LP/Static	LP/Static	LP/Static
PORT0	On	LP/Static	LP/Static	Off	Off
PORT1–4	On	LP/Static	Static	Off	Off
PORT5	On	LP/Static	LP/Static	LP/Static	LP/Static
MICFIL	On	LP/Static	LP/Static	Off	Off
TSI0 Digital	On	LP/Static	LP/Static	Off	Off
TSI0 Analog	On/Off	On/Off	On/Off	Off	Off
Analog modules					
ADC0/1 Digital	On	LP/Static	Static	Off	Off
ADC0/1 Analog	On/Off	On/Off	Off	Off	Off
CMP0/1 Digital	On	LP/Static	LP/Static	LP/Static	Off
CMP0/1 Analog	On/Off	On/Off	On/Off	On/Off	Off
CMP2 Digital	On	LP/Static	LP/Static	Off	Off
CMP2 Analog	On/Off	On/Off	On/Off	On/Off	Off
DAC0/1 Digital	On	LP/Static	Static	Off	Off
DAC0/1 Analog	On/Off	On/Off	On/Off	Off	Off
DAC2 Digital	On	LP/Static	Static	Off	Off
DAC2 Analog	On/Off	On/Off	Off	Off	Off
OPAMP0-2 Digital	On	Static	Optional	Off	Off
OPAMP0-2 Analog	On/Off	On/Off	Off	Off	Off
VREF Digital	On	Static	Static	Off	Off
VREF Analog	On/Off	On/Off	On/Off	Off	Off
SINC filter	On	LP/Static	Static	Off	Off

1. CORE_MAIN and CORE_WAKE subdomains are put into state retention mode.

Table 336. Power domain module operation in low-power modes

Module	Sleep	Deep Sleep	Power Down	Deep Power Down	VBAT
Power modules					
LDO_CORE	Optional LP	On, LP, or off	On, LP, or off	Off	Off

Table continues on the next page...

Table 336. Power domain module operation in low-power modes (continued)

Module	Sleep	Deep Sleep	Power Down	Deep Power Down	VBAT
LDO_SYS	Optional LP	On, LP	On, LP	On, LP	Off
DCDC_CORE	Optional LP	On, LP, or off	On, LP, or off	Off	Off
POR	On	On	On	On	Off
VDD_CORE HVD and LVD	Optional	Optional	Optional	Off	Off
VDD HVD and LVD	Optional	Optional	Optional	Off	Off
VDD_SYS HVD and LVD	Optional	Optional	Optional	Optional	Off

NOTE

You cannot disable the regulator that powers the chip in Power Down mode and in higher-power modes.

35.5 Power supply configurations

35.5.1 Power-efficient supply configuration

Figure 171 shows a power-efficient configuration of the chip power supplies. This configuration uses the on-chip DCDC_CORE switching regulator to power the VDD_CORE supply rail efficiently. For the device package which has VDD_LDO_CORE pin, you should short this VDD_LDO_CORE pin to VDD_CORE pin, and in software you shoud disable LDO_CORE.

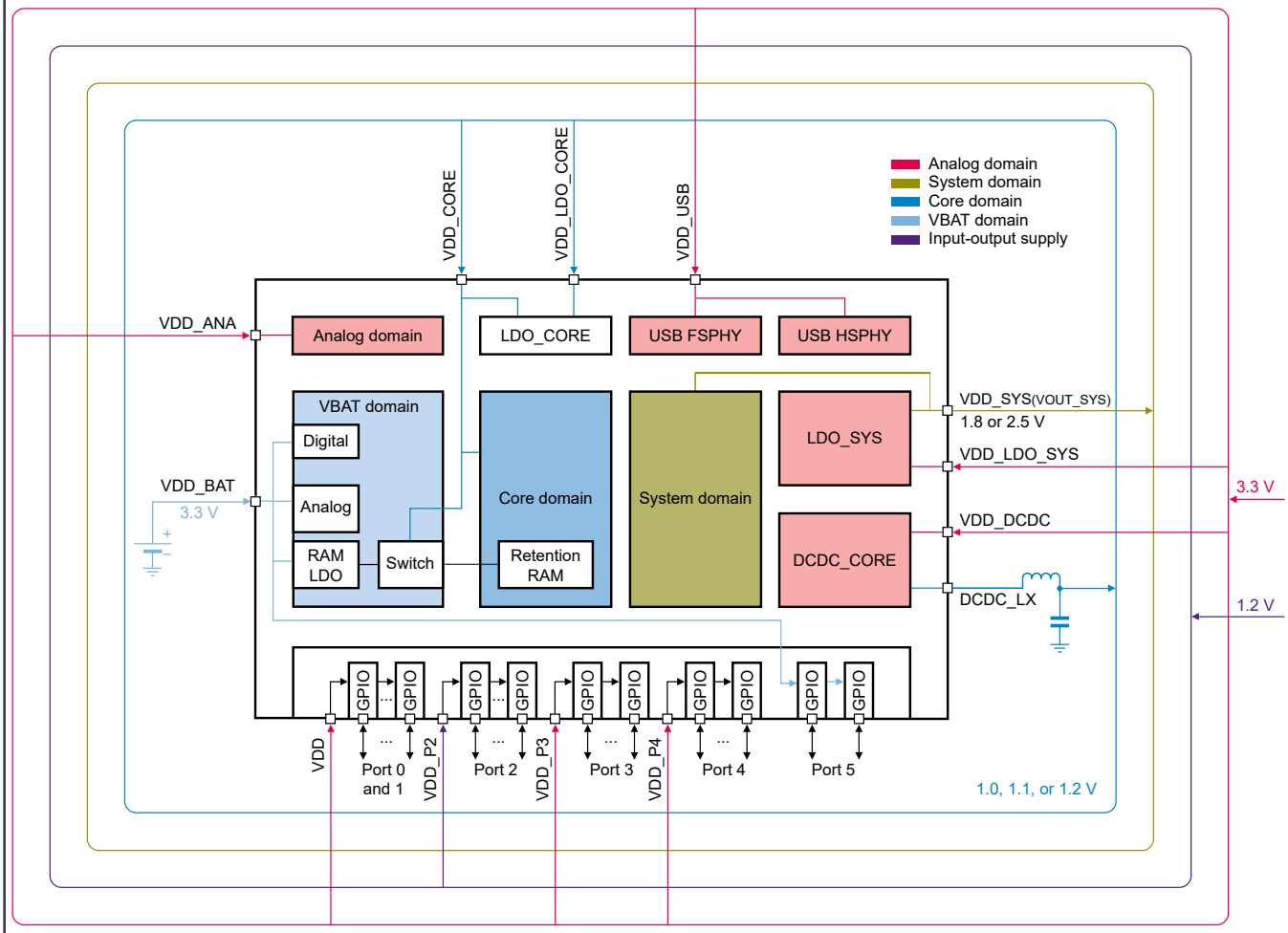


Figure 171. Power-efficient schematic

35.5.2 Low-cost supply configuration

Figure 172 shows a low-cost configuration of the chip power supplies. This configuration uses the on-chip LDO_CORE regulator to save cost and eliminate the passive components for the DCDC_CORE regulator.

For packages where VDD_DCDC has an independent pin, you can connect VDD_DCDC and DCDC_LX to GND with a 10-kΩ resistor to disable DCDC. Your software must disable DCDC.

For packages where VDD_DCDC and VDD_LDO_SYS share a package pin, to disable DCDC, DCDC_LX must be floating, and your software must disable DCDC.

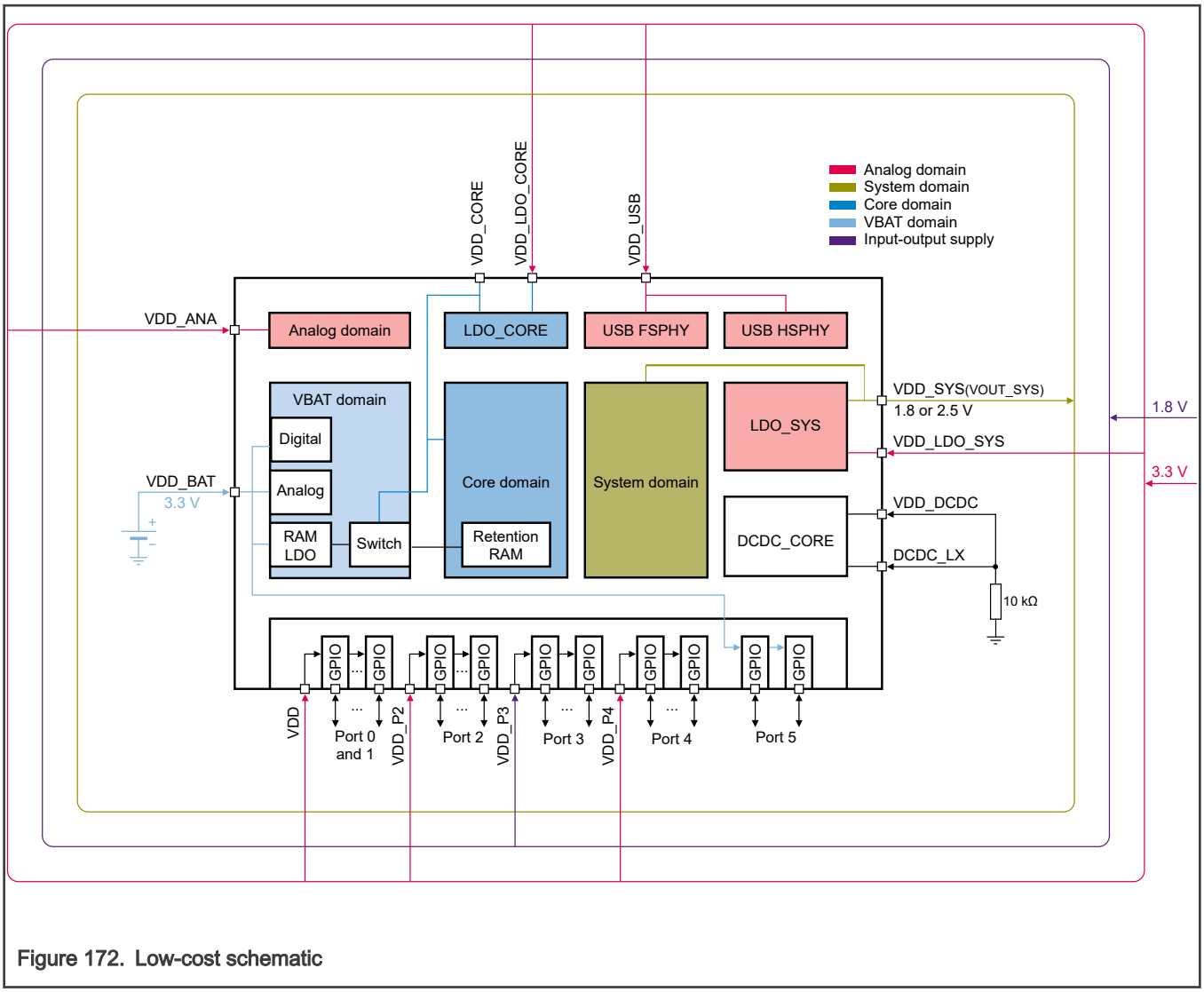


Figure 172. Low-cost schematic

NOTE

See the Hardware Design Guide (UG10092) for more details.

35.6 Power optimization

35.6.1 VDD_CORE voltage scaling

This chip supports VDD_CORE voltage scaling for different performance and power consumption configurations.

This chip supports the following VDD_CORE voltage scaling methods:

- Adjusting VDD_CORE power supply input level directly with Internal Voltage Scaling (IVS) disabled. This method is available for all power modes. To use this method, configure the VDD_CORE voltage level via SPC registers. The high efficiency of the regulators can benefit chip power consumption when DCDC is used as VDD_CORE source. However, the VDD_CORE voltage requires time to reach new levels for charging or discharging the VDD_CORE pin capacitor.
- Using the IVS module. This method applies only to Power Down mode. The VDD_CORE input voltage remains unchanged. The IVS module manipulates the internal supply voltage directly. The voltage reaches the new target value much faster.

Each CORE subdomain can independently enable or disable its own IVS in low-power modes. For example, the CORE_MAIN subdomain can enable its IVS for a lower supply voltage. At the same time, the CORE_WAKE subdomain can continue operating with the higher supply level from VDD_CORE.

35.6.2 SRAM configuration

The SRAM is partitioned, allowing independent configuration of the mode of each SRAM partition.

SRAM power mode	Description
Active	You can access SRAM normally.
Deep Sleep	You cannot access SRAM, but it retains its data.
Shutdown	You cannot access SRAM and it does not retain its data.

Depending on your application requirements, your software can configure an individual SRAM partition into a lower-power mode relative to the overall chip power mode.

When the overall chip power mode is...	You can configure an SRAM partition to operate in this mode...
Active or Sleep	Active
Deep Sleep	Deep Sleep
Power Down	Deep Sleep or Shutdown

NOTE

All SRAM partitions are forced into Shutdown mode after the chip enters Deep Power Down mode, except RAMA. The VBAT domain can power the RAMA partition.

35.6.3 Low-power options for flash memory

The internal flash memory can enter low-power mode when the overall chip is in these modes:

- Active mode under software control (FLASHCR[FLASHDIS])
- Sleep mode, optionally (FLASHCR[FLASHDOZE])
- Deep Sleep mode

NOTE

To avoid unexpected access latencies, take care when configuring the flash module to a low-power mode when the overall chip is in Active mode. The flash module needs time to recover full-speed functionality after entering a low-power mode.

35.6.4 Peripheral clock gating

To conserve power, you can configure the SYSCON module registers to turn off or divide the clocks to modules.

35.6.5 Voltage monitor optimization

Depending on your application requirements, your software can disable individual voltage monitors (LVD, HVD, or both) within the different power domains to save power.

35.6.6 Wake-up time consideration

This chip supports different low-power modes for power consumption level and wake-up time balance.

The wake-up time from Deep Sleep or Power Down mode is the accumulation of:

- VDD_CORE power source recovery time.
- Clock recovery time or flash memory recovery time, whichever is longer.
- Interrupt latency.

The wake-up time from Deep Power Down mode is the boot-up time required after POR releases the chip. This time includes everything in a cold power up, including the boot ROM latency.

These factors can affect the clock recovery time:

- The system clock source status in low-power modes. If the clock remains on in a low-power mode, no recovery time is required.
- The system clock source before entering low-power mode.

The flash memory recovery time depends on the flash memory status in low-power modes.

These factors can affect the VDD_CORE power source recovery time:

- The regulator type used. An on-chip linear regulator starts up faster than an on-chip DCDC.
- The regulator status in low-power modes. If the regulator remains in full-power regulation in low-power modes, no recovery time is required.
- Whether VDD_CORE voltage scaling is enabled before and after entering low-power modes.
- The VDD_CORE voltage scaling method used. Voltage scaling via internal IVS recovers faster.
- If scaling is achieved by adjusting the VDD_CORE voltage directly, the external capacitor on the VDD_CORE pin affects recovery time.

Chapter 36
VBAT

36.1 Chip-specific VBAT information

Table 337. Reference links to related information

Topic	Related module	Reference
Full description	VBAT	VBAT
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

36.1.1 Module instances

This device contains one instance of the VBAT module, VBAT0.

NOTE

VBAT includes some security functionality that is not described here. Refer to the MCX N Security RM for more details.

36.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

36.1.3 WAKEUP_b signal pinout

VBAT_WAKEUP_b is the default signal for P5_2 after a chip reset.

36.1.4 Connections for OSCCLKE[CLKEn] and FROCLKE[CLKEn]

Table 338. Connections for OSCCLKE[CLKEn] and FROCLKE[CLKEn]

VBAT clock enable	Connects to this power domain
CLKE3	CORE_MAIN
CLKE2	CORE_WAKE
CLKE1	VDD_SYS
CLKE0	VDD_BAT

36.1.5 Clearing the CLOCK_DET flag when the CLKMON is disabled

When the CLKMON is disabled, the following register write sequence is needed to clear the STATUSA[CLOCK_DET] register bit:

1. First write 0x5 to the MONTSTA register,
2. Then write 0x0 to MONTSTA.

36.1.6 VBAT power supply

VBAT is supplied from the chip power supply VDD_BAT.

36.1.7 Connections for VBAT interrupt detect signals [IRQn_DET]

Table 339. Connections for [IRQn_DET]

VBAT interrupt detect	Connects to
IRQ3_DET	TDET interrupt
IRQ2_DET	RTC interrupt
IRQ1_DET	GPIO5 interrupt 1
IRQ0_DET	GPIO5 interrupt 0

36.1.8 VBAT LDORAMC[RET] configuration

LDORAMC.RET[3:0] controls retention of RAMA3-RAMA0 in low-power mode in this chip. Each of the four RAMA arrays are 8 KB in size. For example:

- LDORAMC[RET]=0x0 retains all 32 KB of RAMA
- LDORAMC[RET]=0xE retains 8 KB of RAMA0
- LDORAMC[RET]=0xF does not retain any arrays of RAMA

36.2 Overview

VBAT works with the power management system to implement power-saving mechanisms. It provides bandgap timers and a voltage regulator to supply retention SRAM in low-power modes.

VBAT implements the 16 kHz internal clock source.

36.2.1 Block diagram

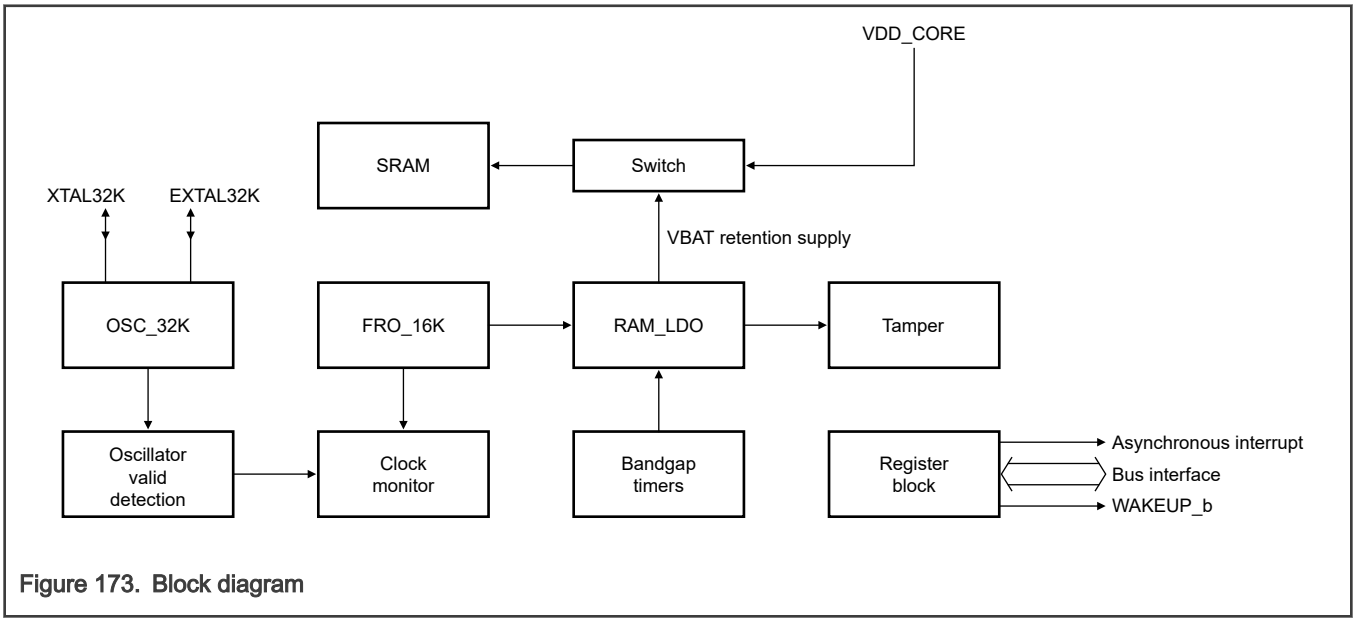


Figure 173. Block diagram

36.2.2 Features

- Includes oscillator for 32.768 kHz crystal (OSC32k)
- Supports internal 16 kHz free-running oscillator (FRO16K)
- Supports backup retention SRAM regulator
- Includes clock monitor (high/low/loss) for the OSC32k clock source
- Supports voltage monitor (high/low) for the VBAT supply
- Monitors temperature (high/low)

36.3 Functional description

The following sections describe the functional details of VBAT.

36.3.1 Operations

This section describes the operations of VBAT.

36.3.1.1 Register protection

The VBAT registers are implemented as separate A and B registers. When configuring an A register, you must write the inverse value to the corresponding B register. The entire B register can be written as the inverse of the A register (for example, LDOCTLB = ~LDOCTLA). The reserved bits in the B register are not writable, and read as 0's. For example, if LDOCTLA = 0x5, LDOCTLB can be written as ~LDOCTLA, and then LDOCTLB will read back as 0x2.

After configuring the A and B registers for a module, it is possible to lock the configuration until the next VBAT POR by configuring the corresponding Lock registers. After the configuration is locked, if any protected A register does not match the inverse of the corresponding B register, then STATUSA[CONFIG_DET] becomes 1. This can trigger an interrupt.

36.3.1.2 FRO 16.384 kHz clock

FRO16K is enabled by default on POR. You can disable it or lock it to prevent any change to the enable field.

36.3.1.3 OSC 32.768 kHz clock

OSC32K is disabled by default on VBAT POR. When enabled, it takes the oscillator start-up time before STATUSA[OSC_RDY] becomes 1, and the clock is output to other modules.

The OSC32k supports 2 modes of operation, a high performance transconductance oscillator mode and a low power switched oscillator mode. A software configurable capacitor bank is supported in normal mode and must be configured when the oscillator is first enabled.

NOTE

For the oscillator characteristics of each mode, refer to the VBAT section of the device datasheet

36.3.1.4 SRAM LDO

SRAM LDO is a retention regulator that retains VBAT retention SRAM in low-power modes, including when VBAT is the only available supply. When IRQENA[LDO_RDY] becomes 1, the chip can enter into a low-power mode where the VBAT supply retains SRAM. Enable the bandgap and LDO before entering a low-power mode where the VBAT supply powers the retention SRAM. Enable Refresh mode for the lowest power consumption.

It takes approximately 8ms for the LDO_RDY flag to set after software enables the bandgap and LDO. When using the SRAM LDO it is recommended to leave it enabled in active modes to avoid this delay before entering a low power mode.

Although SRAM is automatically retained in all low-power modes, you can manually switch SRAM to the VBAT retention supply anytime. Do not access the SRAM when it receives power from the VBAT retention supply. You must manually reverse these steps before accessing the SRAM again.

To manually switch VBAT to power the SRAM:

1. Isolate the SRAM array (write 1 to LDORAMC[ISO]).
2. Switch the supply from VDD_CORE to the VBAT retention LDO (write 1 to LDORAMC[SWI]).

LDORAMC[RETn] can configure how much SRAM is retained in low-power modes. Each bit of the field corresponds to a different SRAM array, and you can configure the number of arrays retained (when SRAM LDO is enabled, retain at least one array).

When SRAM LDO is disabled, LDORAMC[RETn] controls whether the retention of VBAT SRAM happens using VDD_CORE in low-power modes.

In general, using the SRAM LDO to retain the VBAT retention SRAM is a lower power option than using VDD_CORE directly.

NOTE

Enable FRO16K before enabling SRAM LDO or the bandgap.

36.3.1.5 Bandgap timer

Two software-configurable bandgap timers are available when the SRAM LDO bandgap is enabled. These timers share logic with the bandgap timer refresh logic to minimize power, so they use the FRO16K clock source and are only available when the bandgap is enabled. When you change the configured timeout, NXP recommends that you:

1. Disable the timer.
2. Clear the flag.
3. Write the new timeout value.
4. Enable the timeout again.

36.3.1.6 Clock monitor

VBAT has two clock monitors. One uses FRO_16K as its clock source. The other uses OSC_RTC as its clock source. Each monitor checks a divided version of the other clock at the same frequency but divided by 16. The monitor expects a clock edge every 8 cycles, and you can trim the monitor to assert after 10, 12, 14, or 16 cycles with no edge.

NOTE

See the chip's security reference manual for more information on enabling clock monitor and analog tamper features.

36.3.1.7 Analog tamper

You can configure the analog tamper module to generate an interrupt or tamper event using the following detectors:

- VBAT voltage is out of range.
- Temperature is out of range.

NOTE

Enable both the FRO16K and the LDO bandgaps before configuring the analog tamper module.

36.3.2 Low-power modes

VBAT remains functional in all low-power modes.

The VBAT module supports operation in a VBAT only power mode where only the VBAT is powered and the rest of the device is powered off externally. The VBAT_WAKEUP_b output can be used to control an external power switch or to communicate with an

external PMIC.The VBAT_WAKEUP_b output drives low to indicate power supplies to the rest of the device should be powered. This occurs when either WAKECFG[OUT] is low or a status flag as enabled by WAKENA/WAKENB has asserted.When software writes WAKECFG[OUT]=1 and there are no configured status flags asserted, the VBAT_WAKEUP_B output drives high indicating that the rest of the device can be powered off.When a configured status flag asserts when only VBAT is powered, this will cause the VBAT_WAKEUP_b to drive low again and to power up the rest of the device. When WAKEUP_b pin is configured for open drain operation, an external wakeup source can drive this pin low, and the falling edge will assert the WAKEUP_FLAG.

36.3.3 Debug mode

Debug modes do not affect VBAT.

36.3.4 Clocks

VBAT implements the following clock domains:

- Bus interface clock - access the registers
- FRO16 kHz internal clock
- Crystal oscillator 32.768 kHz clock

36.3.5 Reset

Only the POR resets VBAT.

36.3.6 Interrupts

VBAT generates one interrupt combining the following sources:

- VBAT POR
- Wake-up pin assertion
- Bandgap timer 0
- Bandgap timer 1
- Crystal oscillator ready
- Clock monitor frequency error
- VBAT configuration error
- Voltage monitor detect
- Temperature monitor detect

36.4 External signals

Table 340. External signals

Signal	Description	Direction
EXTAL32K	Oscillator input for 32.768 kHz crystal.	Input
XTAL32K	Oscillator output for 32.768 kHz crystal.	Output
WAKEUP_b	Active low wake-up pin, falling edge sets the WAKEUP_FLAG. When WAKEUP_b pin is configured for open drain operation, an external wakeup source can drive this pin low, and the falling edge will assert the WAKEUP_FLAG	Input/Output

36.5 Initialization

To enable and lock the FRO16K:

1. Write 1h to FROCTLA[FRO_EN].
2. Write 0h to FROCTLB[INVERSE].
3. Write 1h to FROLCKA[LOCK].
4. Write 0h FROLCKB[LOCK].
5. Alter FROCLKE[CLKE] to clock gate different FRO16K outputs to different peripherals to reduce power consumption.

To enable and lock the LDO and bandgap:

1. Enable the FRO16K.
2. Write 7h to LDO_RAM Control A (LDOCTLA).
3. Write 0h to LDOCTLB[INVERSE].
4. Wait for STATUSA[LDO_RDY] to become 1.
5. Write 1h to LDOLCKA[LOCK].
6. Write 0h to LDOLCKB[LOCK].

To configure and lock OSC32kHz for normal mode operation:

1. Configure OSCCTLA[EXTAL_CAP_SEL], OSCCTLA[XTAL_CAP_SEL] and OSCCTLA[COARSE_AMP_GAIN] as required based on the external crystal component ESR and CL values, and by the PCB parasitics on the EXTAL32K and XTAL32K pins. Configure 0h to OSCCTLA[MODE_EN], 1h to OSCCTLA[CAP_SEL_EN], and 1h to OSCCTLA[OSC_EN].
2. Write inverse of OSCCTLA to OSCCTLB[INVERSE].
3. Wait for STATUSA[OSC_RDY] to become 1.
4. Write 1h to OSCLCKA[LOCK].
5. Write 0h to OSCLCKB[LOCK].

To configure OSC32kHz for low power mode operation:

1. Write 3h to OSCCFGA[INIT_TRIM].
2. Write inverse of OSCCFGA to OSCCTLB[INVERSE].
3. Configure OSCCTLA[EXTAL_CAP_SEL], OSCCTLA[XTAL_CAP_SEL] and OSCCTLA[COARSE_AMP_GAIN] as required based on the external crystal component ESR and CL values, and by the PCB parasitics on the EXTAL32K and XTAL32K pins. Configure 1h to OSCCTLA[MODE_EN], 1h to OSCCTLA[CAP_SEL_EN], and 1h to OSCCTLA[OSC_EN].
4. Write inverse of OSCCTLA to OSCCTLB[INVERSE].
5. Wait for STATUSA[OSC_RDY] to become 1.
6. Write 0h to OSCCFGA[INIT_TRIM].
7. Write inverse of OSCCFGA to OSCCFGB[INVERSE].
8. Configure 3h to OSCCTLA[MODE_EN], 0h to OSCCTLA[EXTAL_CAP_SEL] and 0h to OSCCTLA[XTAL_CAP_SEL]. Configure OSCCTLA[SUPPLY_DET] as required by application.
9. Write inverse of OSCCTLA to OSCCTLB[INVERSE].
10. Wait for STATUSA[OSC_RDY] to become 1.

36.6 Application information

Configure the VBAT LDO to retain the state of any required SRAM to guard against an unexpected loss of power to the system:

1. Enable the LDO and bandgap.
2. Configure LDORAMC[RETn] to retain the appropriate arrays.

For a software-controlled power down, follow these steps to ensure robust retention of memory before switching off external power:

1. Write 1b to LDORAMC[ISO].
2. Write 1b to LDORAMC[SWI].

You must reverse the above steps to access the SRAM arrays, which the VBAT LDO retains. That is, follow these steps after external power is again switched on:

1. Write 0b to LDORAMC[SWI].
2. Write 0b to LDORAMC[ISO].

36.7 Memory map and register definition

This section includes VBAT memory map and detailed descriptions of all registers.

NOTE

The VBAT registers are reset on VBAT Cold Reset only (VBAT POR).

36.7.1 VBAT register descriptions

36.7.1.1 VBAT memory map

VBAT0 base address: 4005_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0101_000Fh
10h	Status A (STATUSA)	32	RW	0000_0081h
14h	Status B (STATUSB)	32	RW	000F_003Eh
18h	Interrupt Enable A (IRQENA)	32	RW	0000_0000h
1Ch	Interrupt Enable B (IRQENB)	32	RW	000F_FFFFh
20h	Wake-up Enable A (WAKENA)	32	RW	0000_0000h
24h	Wake-up Enable B (WAKENB)	32	RW	000F_FFFFh
38h	Wake-up Configuration (WAKECFG)	32	RW	0000_0000h
100h	Oscillator Control A (OSCCTLA)	32	RW	0000_0000h
104h	Oscillator Control B (OSCCTLB)	32	RW	000F_FFFFh
108h	Oscillator Configuration A (OSCCFGA)	32	RW	0000_0000h
10Ch	Oscillator Configuration B (OSCCFGB)	32	RW	0000_0FFFh
118h	Oscillator Lock A (OSCLCKA)	32	RW	0000_0000h
11Ch	Oscillator Lock B (OSCLCKB)	32	RW	0000_0001h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
120h	Oscillator Clock Enable (OSCCLKE)	32	RW	0000_0000h
200h	FRO16K Control A (FROCTLA)	32	RW	0000_0001h
204h	FRO16K Control B (FROCTLB)	32	RW	0000_0000h
218h	FRO16K Lock A (FROLCKA)	32	RW	0000_0000h
21Ch	FRO16K Lock B (FROLCKB)	32	RW	0000_0001h
220h	FRO16K Clock Enable (FROCLKE)	32	RW	0000_0000h
300h	LDO_RAM Control A (LDOCTLA)	32	RW	0000_0000h
304h	LDO_RAM Control B (LDOCTLB)	32	RW	0000_0007h
318h	LDO_RAM Lock A (LDOLCKA)	32	RW	0000_0000h
31Ch	LDO_RAM Lock B (LDOLCKB)	32	RW	0000_0001h
320h	RAM Control (LDORAMC)	32	RW	0000_0000h
330h	Bandgap Timer 0 (LDOTIMER0)	32	RW	0000_0000h
338h	Bandgap Timer 1 (LDOTIMER1)	32	RW	0000_0000h
700h	Wakeup 0 Register A (WAKEUP0A)	32	RW	0000_0000h
704h	Wakeup 0 Register B (WAKEUP0B)	32	RW	FFFF_FFFFh
708h	Wakeup 0 Register A (WAKEUP1A)	32	RW	0000_0000h
70Ch	Wakeup 0 Register B (WAKEUP1B)	32	RW	FFFF_FFFFh
7F8h	Wakeup Lock A (WAKLCKA)	32	RW	0000_0000h
7FCh	Wakeup Lock B (WAKLCKB)	32	RW	0000_0001h

36.7.1.2 Version ID (VERID)

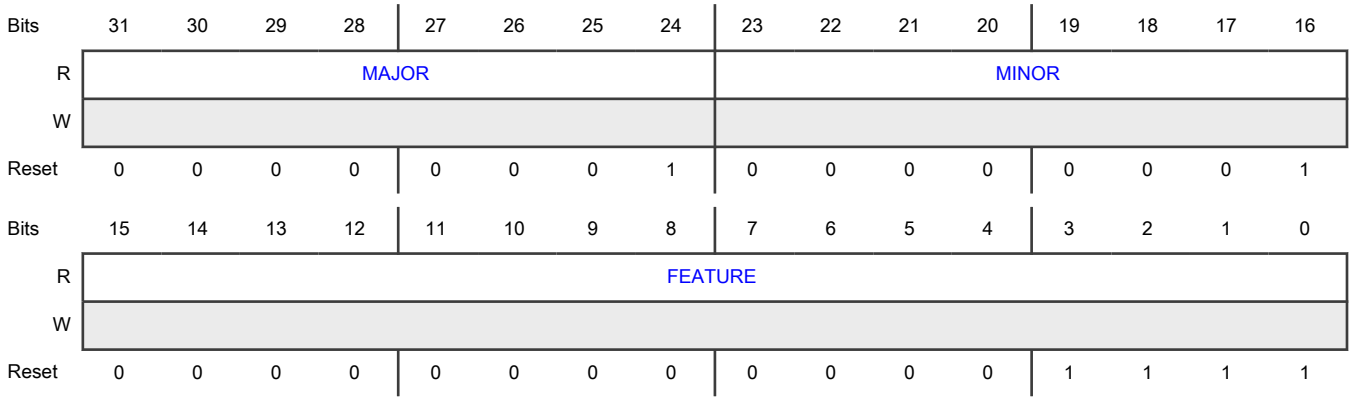
Offset

Register	Offset
VERID	0h

Function

Records the specific version of VBAT in the chip.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the specification.
15-0 FEATURE	Feature Specification Number Returns the feature set number, indicating the feature set present in this instance of VBAT.

36.7.1.3 Status A (STATUSA)

Offset

Register	Offset
STATUSA	10h

Function

Contains all status flags for VBAT.

See the chip's security reference manual for security-related field information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												IRQ3_DET	IRQ2_DET	IRQ1_DET	IRQ0_DET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	CONFIG_...	0	OSC_RDY	LDO_RDY	TIMER1_...	TIMER0_...	WAKEUP_...	POR_DET
W									W1C				W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Fields

Field	Function
31-20 —	Reserved
19 IRQ3_DET	Interrupt 3 Detect Indicates that interrupt 3 has asserted. **NOTE** See the chip-specific VBAT information section for IRQn_DET connections. 0b - Not asserted 1b - Asserted
18 IRQ2_DET	Interrupt 2 Detect Indicates that interrupt 2 has asserted. 0b - Not asserted 1b - Asserted
17 IRQ1_DET	Interrupt 1 Detect Indicates that interrupt 1 has asserted. 0b - Not asserted 1b - Asserted
16 IRQ0_DET	Interrupt 0 Detect Indicates that interrupt 0 has asserted. 0b - Not asserted 1b - Asserted
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7 CONFIG_DET	Configuration Detect Flag Indicates a configuration error in the VBAT registers where one or more A registers are not equal to the inverse of corresponding B registers and the configuration is locked. This may indicate a loss of state or attack. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
6 —	Reserved
5 OSC_RDY	OSC32k Ready Indicates whether OSC32k is enabled and the clock output is stable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled (clock not ready) 1b - Enabled (clock ready)
4 LDO_RDY	LDO Ready Indicates whether LDO is enabled and ready to enter Low-Power mode. 0b - Disabled (not ready) 1b - Enabled (ready)
3 TIMER1_FLAG	Bandgap Timer 1 Flag Asserts periodically according to the bandgap timer 1 period. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not reached 1b - Reached When writing 0b - No effect 1b - Clear the flag
2 TIMER0_FLAG	Bandgap Timer 0 Flag Asserts periodically according to the bandgap timer 0 period. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not reached 1b - Reached When writing 0b - No effect 1b - Clear the flag
1 WAKEUP_FLAG	Wakeup Pin Flag Asserts whenever VBAT detects a falling edge on the wake-up pin. **NOTE** This field behaves differently for register reads and writes. When reading

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not asserted 1b - Asserted When writing 0b - No effect 1b - Clear the flag
0 POR_DET	POR Detect Flag Indicates if the VBAT POR has asserted. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not reset 1b - Reset When writing 0b - No effect 1b - Clear the flag

36.7.1.4 Status B (STATUSB)

Offset

Register	Offset
STATUSB	14h

Function

Contains the inverted status flags for VBAT.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												INVERSE			
W													W0C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INVERSE															
W	W0C															
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0

Fields

Field	Function
31-20 —	Reserved
19-0 INVERSE	Inverse value Contains the inverted contents of Status A (STATUSA).

36.7.1.5 Interrupt Enable A (IRQENA)

Offset

Register	Offset
IRQENA	18h

Function

Contains all interrupt enables for VBAT. When configuring the IRQ enables, both enable A and enable B registers need to be programmed to the appropriate values.

NOTE

See the chip's security reference manual for security-related field information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												IRQ3_DET	IRQ2_DET	IRQ1_DET	IRQ0_DET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	CONFIG_...	0	OSC_RDY	LDO_RDY	TIMER1_...	TIMER0_...	WAKEUP_...	POR_DET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19	Interrupt 3 Detect

Table continues on the next page...

Table continued from the previous page...

Field	Function
IRQ3_DET	Enables the corresponding interrupt in Status A (STATUSA). **NOTE** See the chip-specific VBAT information section for IRQn_DET connections. 0b - Disable 1b - Enable
18 IRQ2_DET	Interrupt 2 Detect Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
17 IRQ1_DET	Interrupt 1 Detect Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
16 IRQ0_DET	Interrupt 0 Detect Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
15 —	Reserved
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10 —	Reserved
9	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
8 —	Reserved
7 CONFIG_DET	Configuration Detect Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
6 —	Reserved
5 OSC_RDY	OSC32k Ready Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
4 LDO_RDY	LDO Ready Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
3 TIMER1_FLAG	Bandgap Timer 2 Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
2 TIMER0_FLAG	Bandgap Timer 0 Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
1 WAKEUP_FLAG	Wakeup Pin Flag Enables the corresponding interrupt in Status A (STATUSA). 0b - Disable 1b - Enable
0	POR Detect Enables the corresponding interrupt in Status A (STATUSA).

Table continues on the next page...

Table continued from the previous page...

Field	Function
POR_DET	0b - Disable 1b - Enable

36.7.1.6 Interrupt Enable B (IRQENB)

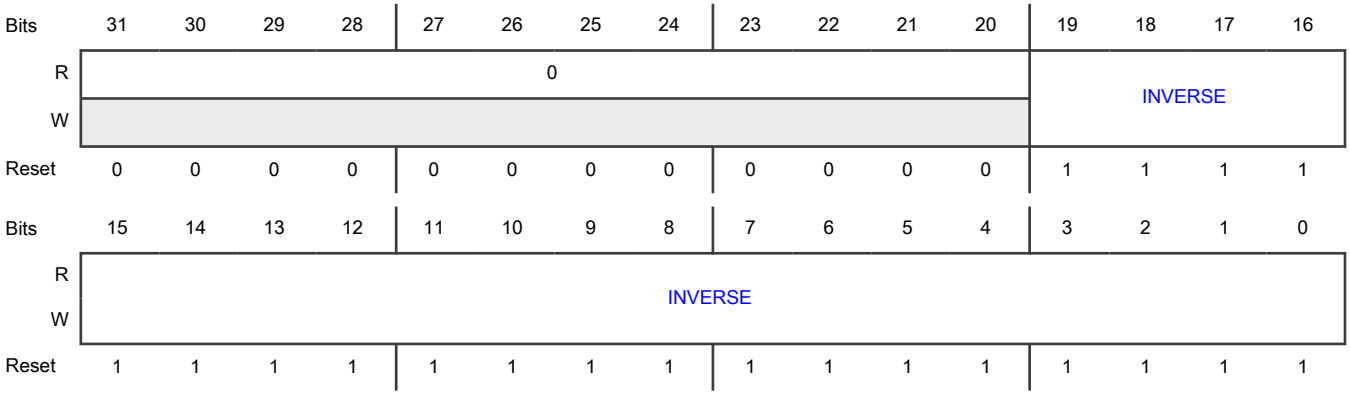
Offset

Register	Offset
IRQENB	1Ch

Function

Contains the inverted interrupt enables for VBAT.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-0 INVERSE	Inverse Value Contains the inverted contents of Interrupt Enable A (IRQENA). When configuring the IRQ enables, both enable A and enable B registers need to be programmed to the appropriate values.

36.7.1.7 Wake-up Enable A (WAKENA)

Offset

Register	Offset
WAKENA	20h

Function

Contains all wake-up enables for VBAT. When configuring the wake enables, both enable A and enable B registers need to be programmed to the appropriate values.

NOTE

See the chip's security reference manual for security-related field information

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												IRQ3_DET	IRQ2_DET	IRQ1_DET	IRQ0_DET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	CONFIG_...	0	OSC_RDY	LDO_RDY	TIMER1_...	TIMER0_...	WAKEUP_...	POR_DET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19 IRQ3_DET	Interrupt 3 Detect Enables the corresponding wake-up in Status A (STATUSA). **NOTE** See the chip-specific VBAT information section for IRQn_DET connections. 0b - Disable 1b - Enable
18 IRQ2_DET	Interrupt 2 Detect Enables the corresponding wake-up in Status A (STATUSA).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
17 IRQ1_DET	Interrupt 1 Detect Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable
16 IRQ0_DET	Interrupt 0 Detect Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable
15 —	Reserved
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7 CONFIG_DET	Configuration Detect Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 —	Reserved
5 OSC_RDY	OSC32K Ready Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable
4 LDO_RDY	LDO Ready Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable
3 TIMER1_FLAG	Bandgap Timer 2 Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable
2 TIMER0_FLAG	Bandgap Timer 0 Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable
1 WAKEUP_FLAG	Wake-up Pin Flag Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable
0 POR_DET	POR Detect Enables the corresponding wake-up in Status A (STATUSA). 0b - Disable 1b - Enable

36.7.1.8 Wake-up Enable B (WAKENB)

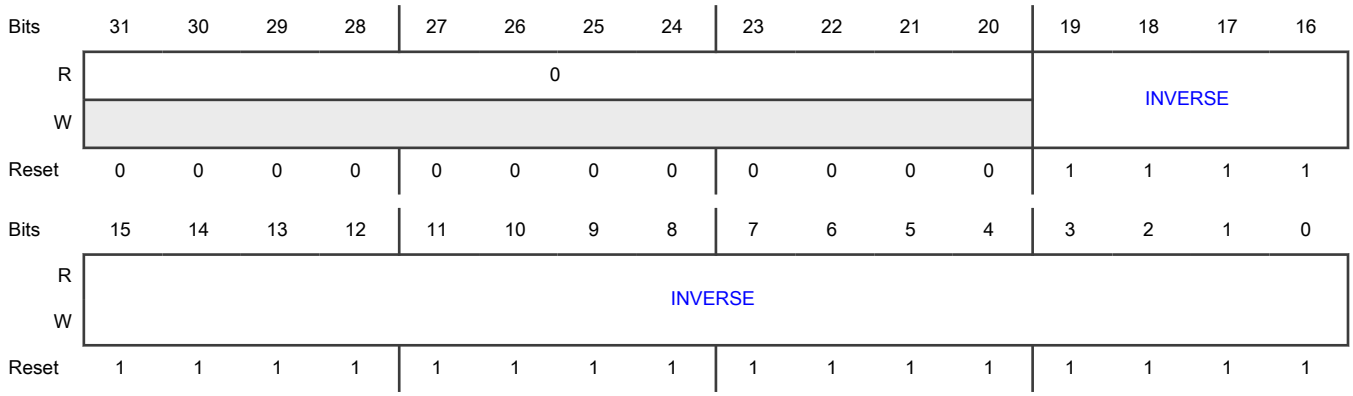
Offset

Register	Offset
WAKENB	24h

Function

Contains the inverted wake-up enables for VBAT. When configuring the wake enables, both enable A and enable B registers need to be programmed to the appropriate values.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-0 INVERSE	Inverse Value Contains the inverted contents of Wake-up Enable A (WAKENA).

36.7.1.9 Wake-up Configuration (WAKECFG)

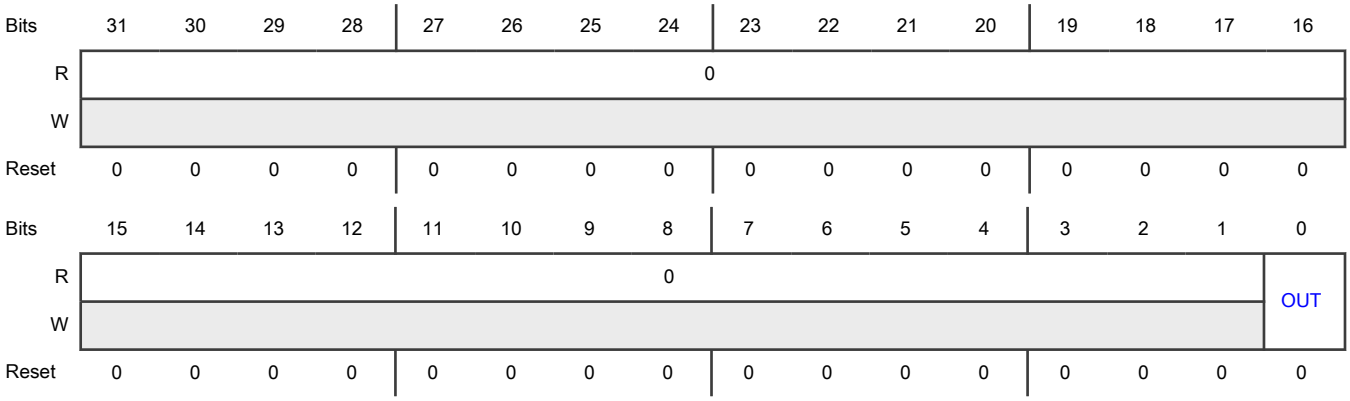
Offset

Register	Offset
WAKECFG	38h

Function

Configures the default state of the WAKEUP_b signal when no enabled wake-up source is asserted.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 OUT	Output Specifies the output state of WAKEUP_b signal. When the value of this field is 1, WAKEUP_b output state is logic one, unless an enabled wakeup source asserts. 0b - Logic zero (asserted) 1b - Logic one

36.7.1.10 Oscillator Control A (OSCCTLA)

Offset

Register	Offset
OSCCTLA	100h

Function

Contains the oscillator control fields. When configuring the oscillator, both control A and control B registers need to be programmed to the appropriate values. Writes to this register are blocked when OSCLCKA[LOCK] is 1.

Some fields in this register uses the following equation to calculate the load capacitance:

$$C_L = \frac{C_{extal} \times C_{xtal}}{C_{extal} + C_{xtal}} + C_{shunt}$$

Equation 16. Equation for load capacitance

where:

- C_{extal} is the selected capacitance on the EXTAL pin.
- C_{xtal} is the selected capacitance on the XTAL pin.
- C_{shunt} is the internal shunt capacitance. You can determine this capacitance value from the chip's data sheet.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												SUPPLY_DET		MODE_EN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	XTAL_CAP_SEL				EXTAL_CAP_SEL				CAP_SEL...	Reserved	Reserved		COARSE_AMP_GAIN		OSC_BYP...	OSC_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-18 SUPPLY_DET	Supply Detector Trim Specifies supply detector for low power mode. 00b - VBAT supply is less than 3V 01b - VBAT supply is greater than 3V
17-16 MODE_EN	Mode Enable Configures Crystal Oscillator mode. 00b - Normal mode 01b - Startup mode 11b - Low power mode
15-12 XTAL_CAP_SEL	Crystal Load Capacitance Selection Selects the internal capacitance for the XTAL pin from the capacitor bank. Calculate the final load capacitance as shown in Equation 16. You must write 0000b to this field in low power mode. 0000b - 0 pF 0001b - 2 pF 0010b - 4 pF

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0011b - 6 pF 0100b - 8 pF 0101b - 10 pF 0110b - 12 pF 0111b - 14 pF 1000b - 16 pF 1001b - 18 pF 1010b - 20 pF 1011b - 22 pF 1100b - 24 pF 1101b - 26 pF 1110b - 28 pF 1111b - 30 pF
11-8 EXTAL_CAP_SEL	Crystal Load Capacitance Selection Selects the internal capacitance for the EXTAL pin from the capacitor bank. Calculate the final load capacitance as shown in Equation 16. The configuration EXTAL_CAP_SEL=0000 and CAP_SEL_EN=1 is required in low power mode and is not supported in other modes 0000b - 0 pF 0001b - 2 pF 0010b - 4 pF 0011b - 6 pF 0100b - 8 pF 0101b - 10 pF 0110b - 12 pF 0111b - 14 pF 1000b - 16 pF 1001b - 18 pF 1010b - 20 pF 1011b - 22 pF 1100b - 24 pF 1101b - 26 pF 1110b - 28 pF 1111b - 30 pF

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 CAP_SEL_EN	Crystal Load Capacitance Selection Enable Enables the internal capacitor banks on the EXTAL and XTAL pins that vary the load capacitance on these pins. You must write 1 to this field and OSCCTLA[OSC_EN] simultaneously. **NOTE** The configuration EXTAL_CAP_SEL=0000 and CAP_SEL_EN=1 is required in low power mode and is not supported in other modes 0b - Disable 1b - Enable
6 —	Reserved
5-4 —	Reserved
3-2 COARSE_AMP_GAIN	Amplifier gain adjustment bits to allow the use of a wide range of external crystal ESR values See the device datasheet for the ranges supported by this device Tunes the internal transconductance (g_m) by increasing the current. 00b - ESR Range 0 01b - ESR Range 1 10b - ESR Range 2 11b - ESR Range 3
1 OSC_BYP_EN	Crystal Oscillator Bypass Enable Bypasses the crystal oscillator. The oscillator outputs the clock, but this clock is the same as the clock provided on EXTAL pin. You must write 1 to both OSCCTLA[OSC_EN] and this field to enable the Bypass mode of the oscillator. To exit Bypass mode, you must write 0 to both OSCCTLA[OSC_EN] and this field. Do not move the oscillator from Bypass mode directly to Normal mode. 0b - Does not bypass 1b - Bypass
0 OSC_EN	Crystal Oscillator Enable Enables the crystal oscillator. The oscillator starts giving the clock output but the clock is not available to the chip until STATUSA[OSC_RDY] is asserted. 0b - Disable 1b - Enable

36.7.1.11 Oscillator Control B (OSCCTLB)

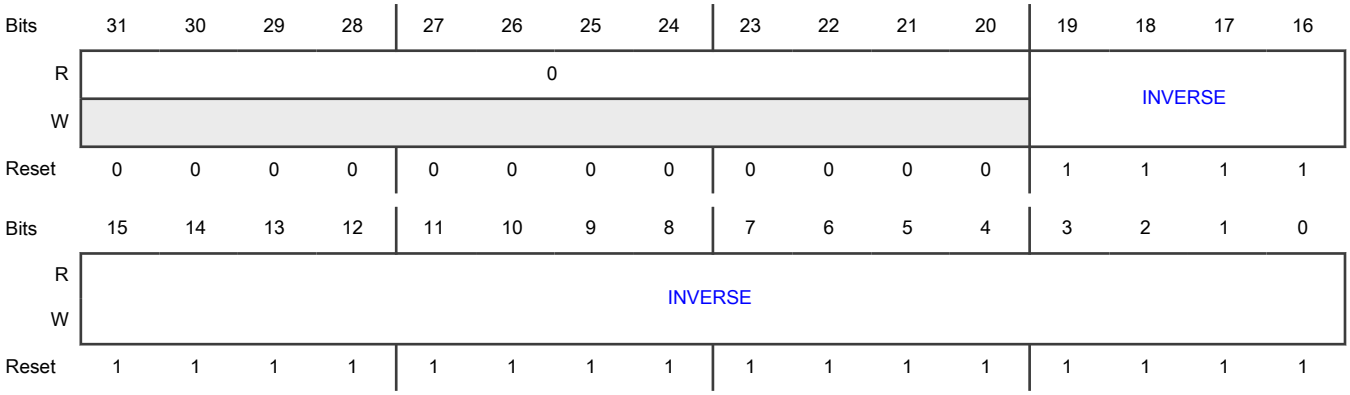
Offset

Register	Offset
OSCCTLB	104h

Function

Contains the inverted oscillator control field. Configure control A and control B registers with the appropriate values when configuring the oscillator. Writes to this register are blocked when OSCLCKB[LOCK] is 1.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-0 INVERSE	Inverse Value Contains the inverted contents of Oscillator Control A (OSCCTLA).

36.7.1.12 Oscillator Configuration A (OSCCFGA)

Offset

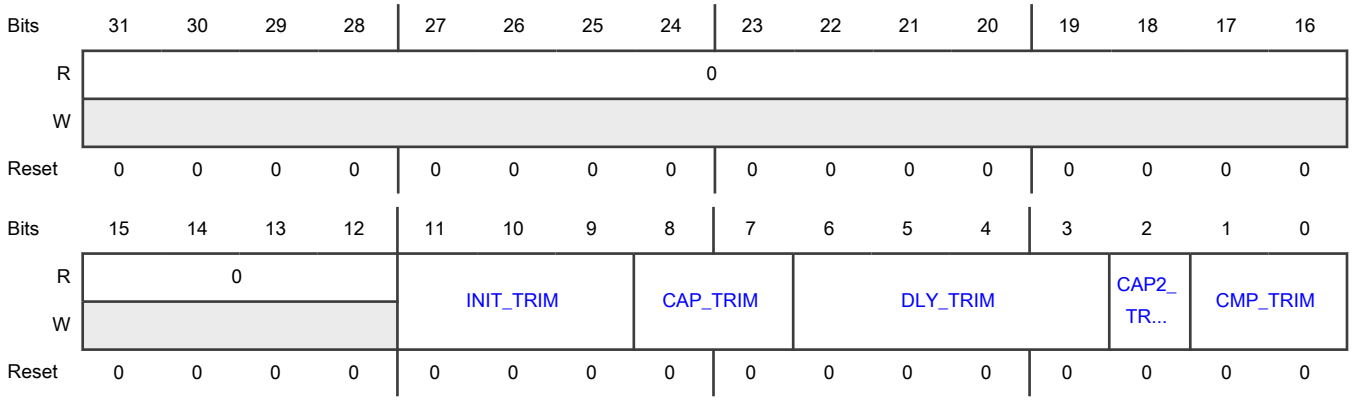
Register	Offset
OSCCFGA	108h

Function

Contains the oscillator configuration fields. When configuring the oscillator, both configuration A and configuration B registers need to be programmed to the appropriate values. Writes to this register are blocked when OSCLCKA[LOCK] is 1.

Reset values are loaded out of IFR.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-9 INIT_TRIM	Initialization Trim Configures the start-up time of the oscillator. 000b - 8 s 001b - 4 s 010b - 2 s 011b - 1 s 100b - 0.5 s 101b - 0.25 s 110b - 0.125 s 111b - 0.5 ms
8-7 CAP_TRIM	Capacitor Trim Cap_trim changes the position of charge injection and removal for low power mode. 00b - Default (when CAP2_TRIM = 0 and CAP_TRIM[1:0] = 00) 01b - -1us (when CAP2_TRIM = 0 and CAP_TRIM[1:0] = 01) 10b - -2us (when CAP2_TRIM = 0 and CAP_TRIM[1:0] = 10) or or +3.5us (when CAP2_TRIM = 1 and CAP_TRIM[1:0] = 10) 11b - -2.5us (when CAP2_TRIM = 0 and CAP_TRIM[1:0] = 11) or +1us (when CAP2_TRIM = 1 and CAP_TRIM[1:0] = 11)
6-3 DLY_TRIM	Delay Trim Changes the p current and n current used for biasing the amplifier, applicable for low power and normal mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000b - P current 9(nA) and N Current 6(nA) 0001b - P current 13(nA) and N Current 6(nA) 0011b - P current 4(nA) and N Current 6(nA) 0100b - P current 9(nA) and N Current 4(nA) 0101b - P current 13(nA) and N Current 4(nA) 0111b - P current 4(nA) and N Current 4(nA) 1000b - P current 9(nA) and N Current 2(nA) 1001b - P current 13(nA) and N Current 2(nA) 1011b - P current 4(nA) and N Current 2(nA)
2 CAP2_TRIM	CAP2_TRIM Affects the function of CAP_TRIM, see CAP_TRIM for details.
1-0 CMP_TRIM	Comparator Trim CMP_TRIM[1:0] changes the internal low power supply for low power and normal mode. 00b - 760 mV 01b - 770 mV 11b - 740 mV

36.7.1.13 Oscillator Configuration B (OSCCFGB)

Offset

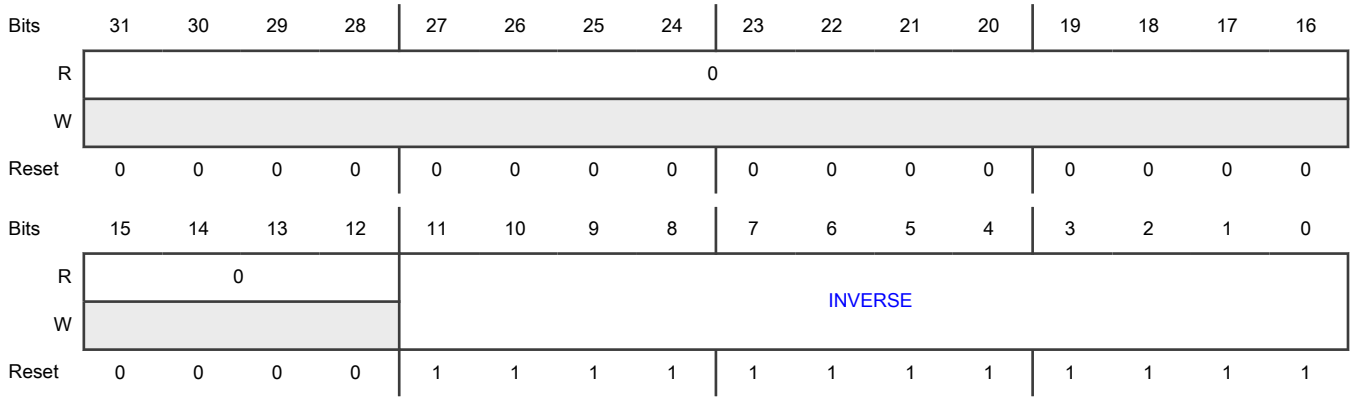
Register	Offset
OSCCFGB	10Ch

Function

Contains the inverted oscillator control field. Configure configuration A and configuration B registers with the appropriate values when configuring the oscillator. Writes to this register are blocked when OSCLCKB[LOCK] is 1.

Reset values are loaded out of IFR.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-0 INVERSE	Inverse Value Contains the inverted contents of Oscillator Configuration A (OSCCFGA).

36.7.1.14 Oscillator Lock A (OSCLCKA)

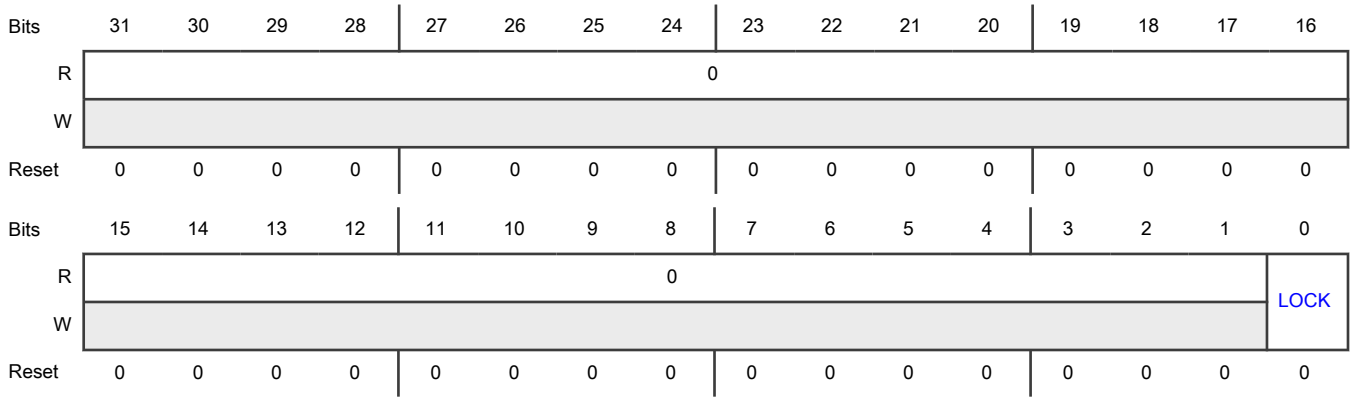
Offset

Register	Offset
OSCLCKA	118h

Function

Allows you to block any write to the oscillator registers.When configuring the oscillator, both lock A and lock B registers need to be programmed to the appropriate values. Writes to this register are blocked when OSCLCKA[LOCK] is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 LOCK	Lock Blocks any write to the oscillator registers. 0b - Do not block 1b - Block

36.7.1.15 Oscillator Lock B (OSCLCKB)

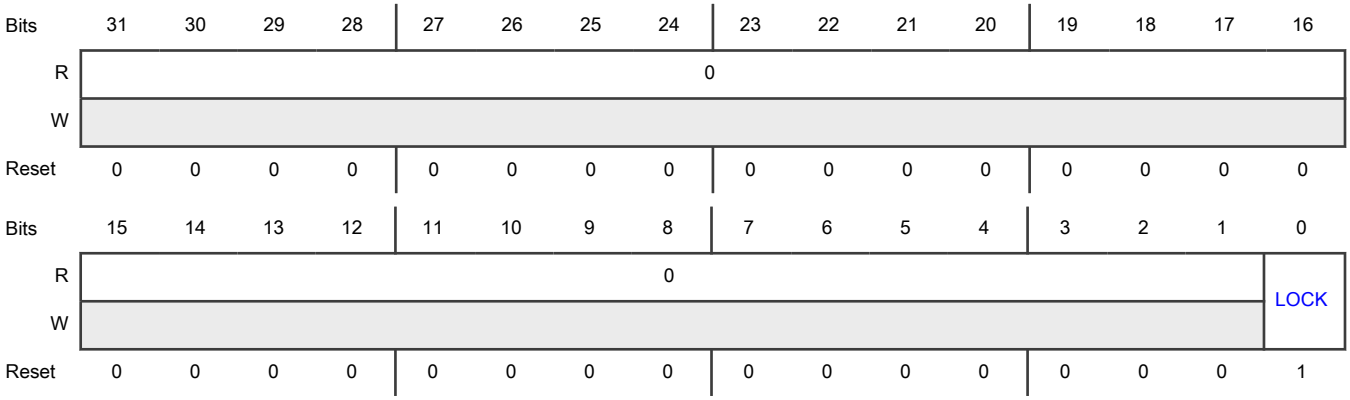
Offset

Register	Offset
OSCLCKB	11Ch

Function

Contains the inverted oscillator lock field. Configure lock A and lock B registers with the appropriate values when configuring the oscillator. Writes to this register are blocked when OSCLCKB[LOCK] is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved
0	Lock

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOCK	Blocks any write to the oscillator registers and asserts a configuration error. The lock is enabled if any of the oscillator A registers are not equal to inverted oscillator B registers. 0b - Block 1b - Do not block

36.7.1.16 Oscillator Clock Enable (OSCCLKE)

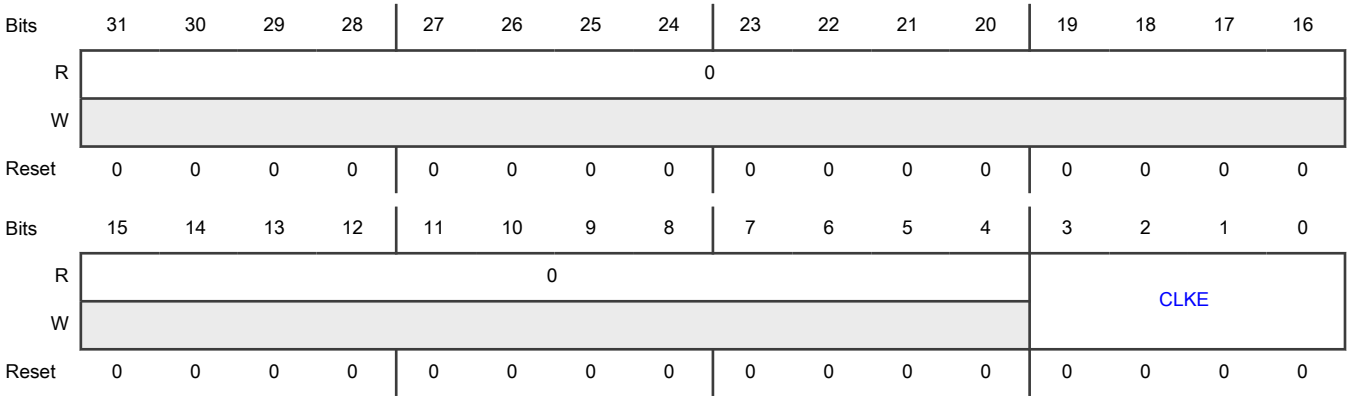
Offset

Register	Offset
OSCCLKE	120h

Function

Contains clock gating register field to gate the OSC32k clock to other modules.

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 CLKE	Clock Enable Enables the corresponding OSC32 kHz output clock to other modules when you write 1 to this field. See the chip-specific VBAT information section for more information.

36.7.1.17 FRO16K Control A (FROCTLA)

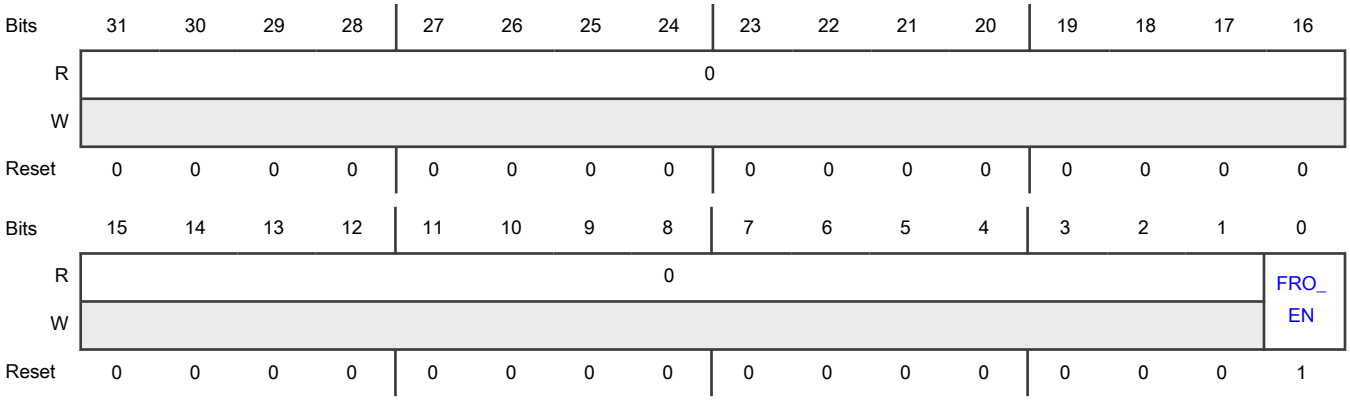
Offset

Register	Offset
FROCTLA	200h

Function

Controls FRO16K. When configuring the FRO16K, both control A and control B registers need to be programmed to the appropriate values. Writes to this register are blocked when FROLCKA[LOCK] is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 FRO_EN	FRO16K Enable 0b - Disable 1b - Enable

36.7.1.18 FRO16K Control B (FROCTLB)

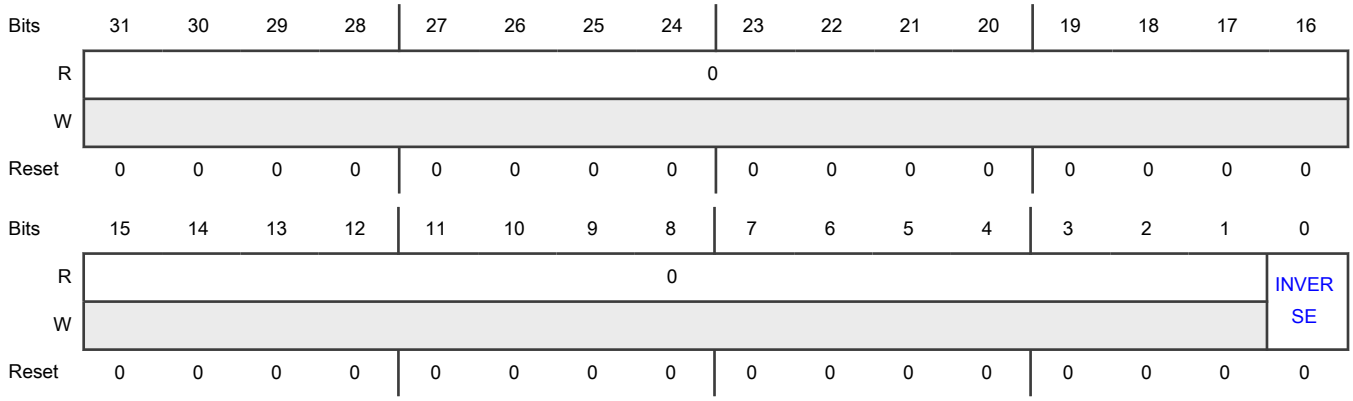
Offset

Register	Offset
FROCTLB	204h

Function

Contains the inverted FRO16K control field. Configure control A and control B registers with the appropriate values when configuring FRO16K. Writes to this register are blocked when FROLCKB[LOCK] is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 INVERSE	Inverse Value Contains the inverted contents of FRO16K Control A (FROCTLA).

36.7.1.19 FRO16K Lock A (FROLCKA)

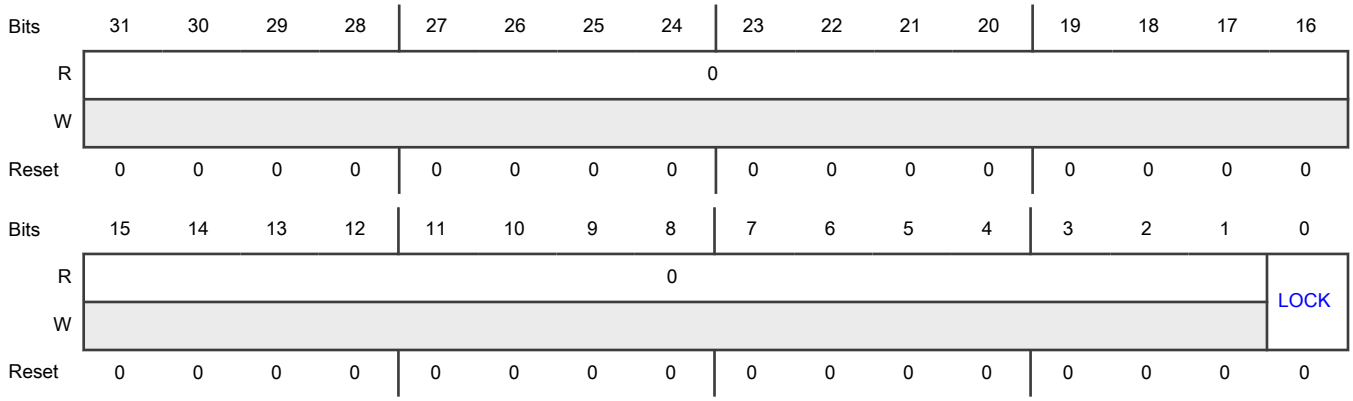
Offset

Register	Offset
FROLCKA	218h

Function

Contains the lock field. When configuring the FRO16K, both lock A and lock B registers need to be programmed to the appropriate values. Writes to this register are blocked when FROLCKA[LOCK] is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 LOCK	Lock Blocks any write to the FRO16K registers when you write 1 to this field. VBAT POR writes 0 to this field. 0b - Do not block 1b - Block

36.7.1.20 FRO16K Lock B (FROLCKB)

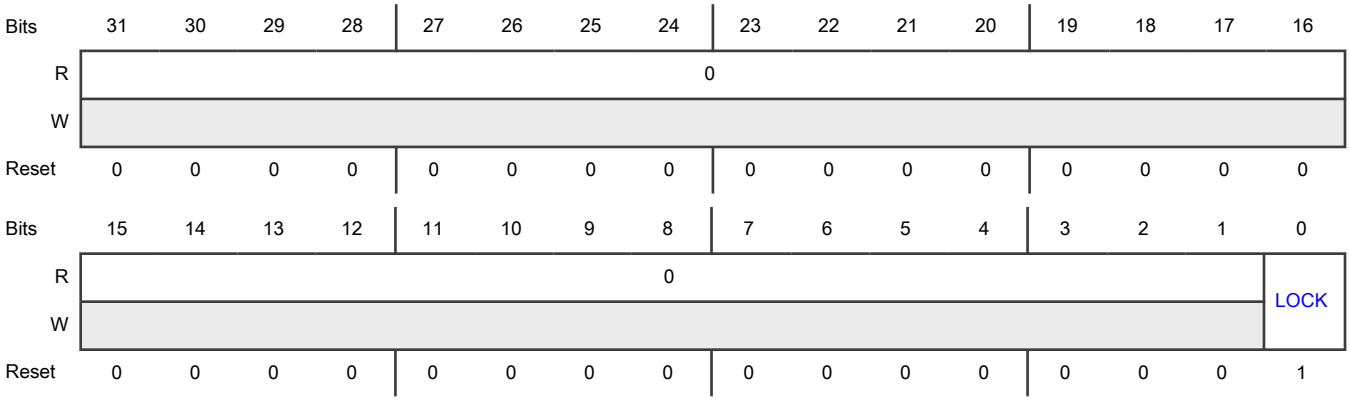
Offset

Register	Offset
FROLCKB	21Ch

Function

Contains the inverted FRO16K lock field. Configure lock A and lock B registers with the appropriate values when configuring FRO16K. Writes to this register are blocked when FROLCKB[LOCK] is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 LOCK	Lock Blocks any write to the FRO16K registers and asserts a configuration error when you write 0 to this field. The lock is enabled if any of the FRO16K A registers are not equal to inverted FRO16K B registers. 0b - Block 1b - Do not block

36.7.1.21 FRO16K Clock Enable (FROCLKE)

Offset

Register	Offset
FROCLKE	220h

Function

Contains clock gating register field to gate the FRO16K clock to other modules.

NOTE

FROCLKE cannot be locked (not affected by FRO16K Lock A (FROLCKA)).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CLKE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-4 —	Reserved
3-0 CLKE	Clock Enable Enables the corresponding FRO16 kHz output clock to other modules when you write 1 to the corresponding bit in this field. See the chip-specific VBAT information section for more information.

36.7.1.22 LDO_RAM Control A (LDOCTLA)

Offset

Register	Offset
LDOCTLA	300h

Function

Contains the LDO_RAM control field. When configuring the LDO_RAM, both control A and control B registers need to be programmed to the appropriate values. Writes to this register are blocked when LDOLCKA[LOCK] is 1.

NOTE

The FRO16K must be enabled before enabling the SRAM LDO or the bandgap

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													REFRESH...	LDO_EN	BG_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-3 —	Reserved
2 REFRESH_EN	Refresh Enable Enables the Bandgap Low-Power Refresh mode. The refresh mode must also be enabled for lowest power consumption. 0b - Refresh mode is disabled 1b - Refresh mode is enabled for low power operation
1 LDO_EN	LDO Enable Enables the backup SRAM regulator. LDOCTLA[BG_EN] must enable the bandgap when the backup SRAM regulator is enabled. After you enable the bandgap and backup SRAM regulator, you must wait until STATUSA[LDO_RDY] becomes 1 before the SRAM can enter a low-power state where the regulator supplies the array contents.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
0 BG_EN	Bandgap Enable Enables the LDO_RAM bandgap. 0b - Disable 1b - Enable

36.7.1.23 LDO_RAM Control B (LDOCTLB)

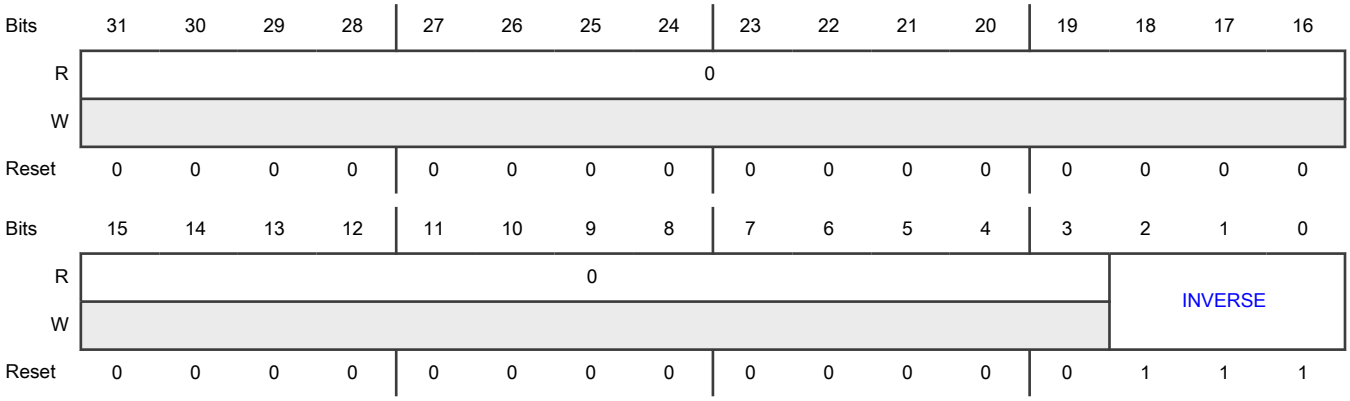
Offset

Register	Offset
LDOCTLB	304h

Function

Contains the inverted LDO_RAM control field. Configure control A and control B registers with the appropriate values when configuring the LDO_RAM. Writes to this register are blocked when LDOLCKB[LOCK] is 1.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 INVERSE	Inverse Value Contains the inverted contents of LDO_RAM Control A (LDOCTLA).

36.7.1.24 LDO_RAM Lock A (LDOLCKA)

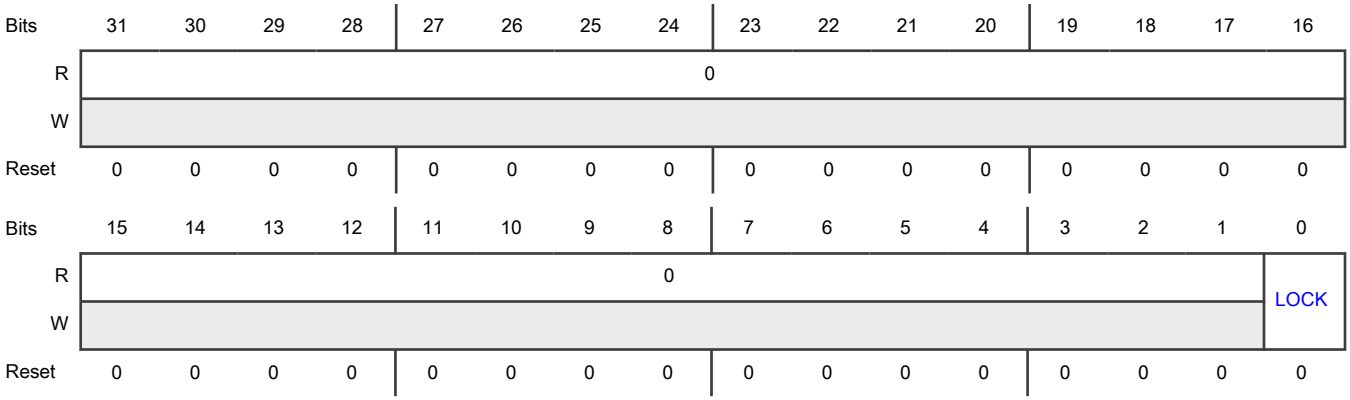
Offset

Register	Offset
LDOLCKA	318h

Function

Contains the lock field. When configuring the LDO_RAM, both lock A and lock B registers need to be programmed to the appropriate values. Writes to this register are blocked when the LDO_RAM lock register is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 LOCK	Lock Blocks any write to the LDO_RAM registers. VBAT POR writes 0 to this field. 0b - Do not block 1b - Block

36.7.1.25 LDO_RAM Lock B (LDOLCKB)

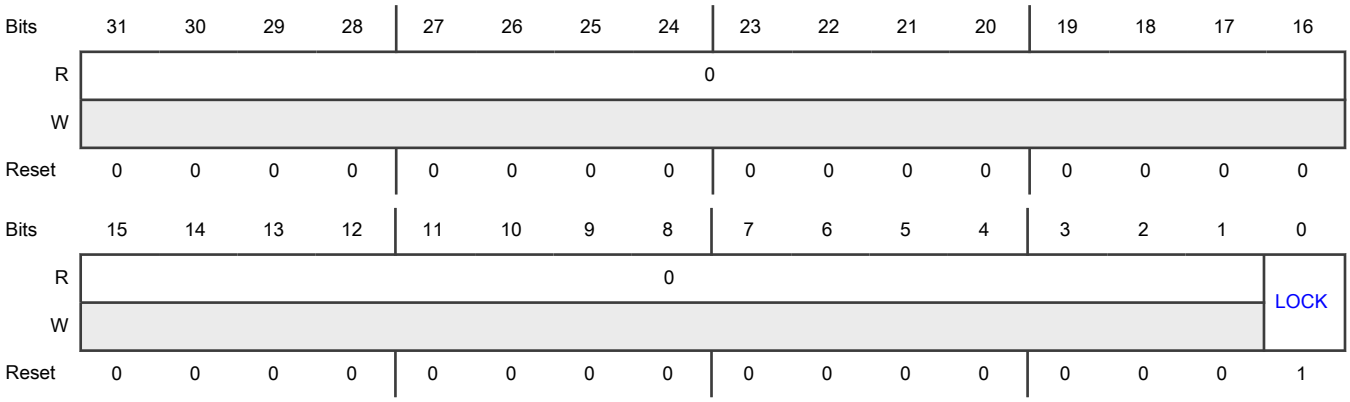
Offset

Register	Offset
LDOLCKB	31Ch

Function

Contains the inverted LDO_RAM lock field. Configure lock A and lock B registers with the appropriate values when configuring the LDO_RAM. Writes to this register are blocked when LDOLCKB[LOCK] is 1.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 LOCK	Lock Blocks any write to the LDO_RAM registers and asserts a configuration error. The lock is enabled if any of the LDO_RAM A registers are not equal to inverted LDO_RAM B registers. 0b - Block 1b - Do not block

36.7.1.26 RAM Control (LDORAMC)

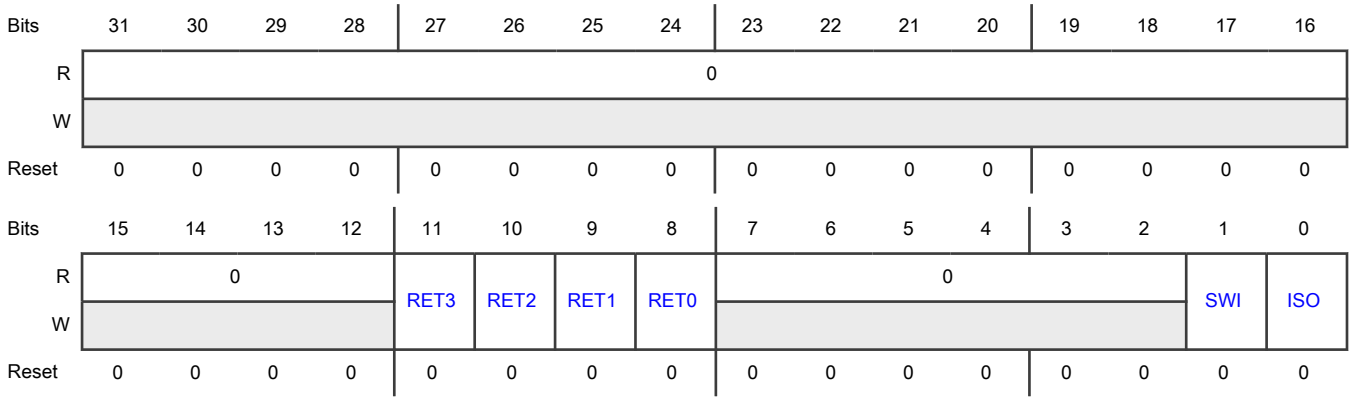
Offset

Register	Offset
LDORAMC	320h

Function

Contains software overrides for the SRAM isolation and backup switch. Always isolate the SRAM array before asserting or negating the switch control.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-8 RETn	Retention Configures retention of each SRAM array in low power modes. See the VBAT chip-specific section for more details on these SRAM arrays. When LDO_RAM is enabled, at least one SRAM array must be retained. When LDO_RAM is disabled, this field specifies whether VDD_CORE continues to provide power to the SRAM array. 0b - Corresponding SRAM array is retained in low-power modes 1b - Corresponding SRAM array is not retained in low-power modes
7-2 —	Reserved
1 SWI	Switch SRAM Specifies how the SRAM array is powered. Change this field when LDORAMC[ISO] = 1. 0b - Supply follows the chip power modes 1b - LDO_RAM powers the array
0 ISO	Isolate SRAM Specifies how the SRAM array is isolated. 0b - State follows the chip power modes 1b - Isolates SRAM and places it in Low-Power Retention mode

36.7.1.27 Bandgap Timer 0 (LDOTIMER0)

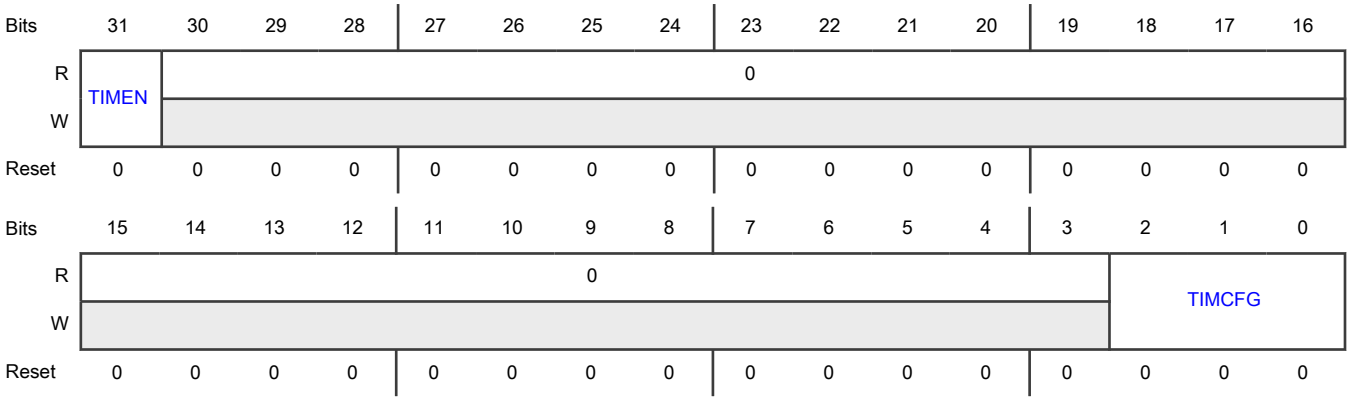
Offset

Register	Offset
LDOTIMER0	330h

Function

Controls one software-configurable timer when the bandgap is enabled and are clocked by FRO16K.

Diagram



Fields

Field	Function
31 TIMEN	Bandgap Timeout Period Enable 0b - Disable 1b - Enable
30-3 —	Reserved
2-0 TIMCFG	Timeout Configuration Configures the bandgap timeout 0 period. Changed when the timer is disabled. 000b - 1 s 001b - 500 ms 010b - 250 ms 011b - 125 ms 100b - 62.5 ms 101b - 31.25 ms 110b - 15.625 ms 111b - 7.8125 ms

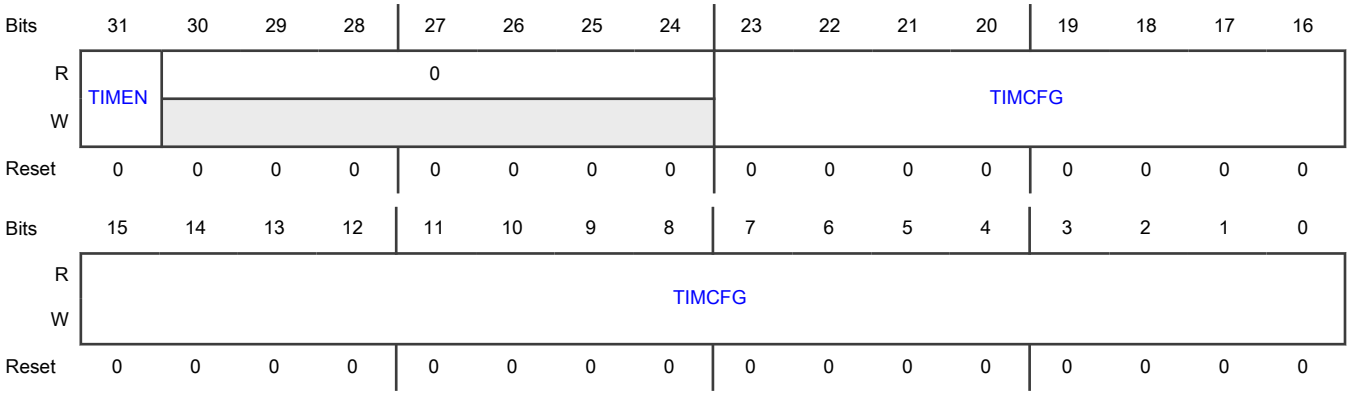
36.7.1.28 Bandgap Timer 1 (LDOTIMER1)

Offset

Register	Offset
LDOTIMER1	338h

Function
Controls the other software-configurable timer when the bandgap is enabled and are clocked by the FRO16K.

Diagram



Fields

Field	Function
31 TIMEN	Bandgap Timeout Period Enable 0b - Disable 1b - Enable
30-24 —	Reserved
23-0 TIMCFG	Timeout Configuration Configures the bandgap timeout 1 period. Configures timeout in number of seconds, ranging from 0 to 65,535 s. You can change this field when the timer is disabled.

36.7.1.29 Wakeup 0 Register A (WAKEUP0A - WAKEUP1A)

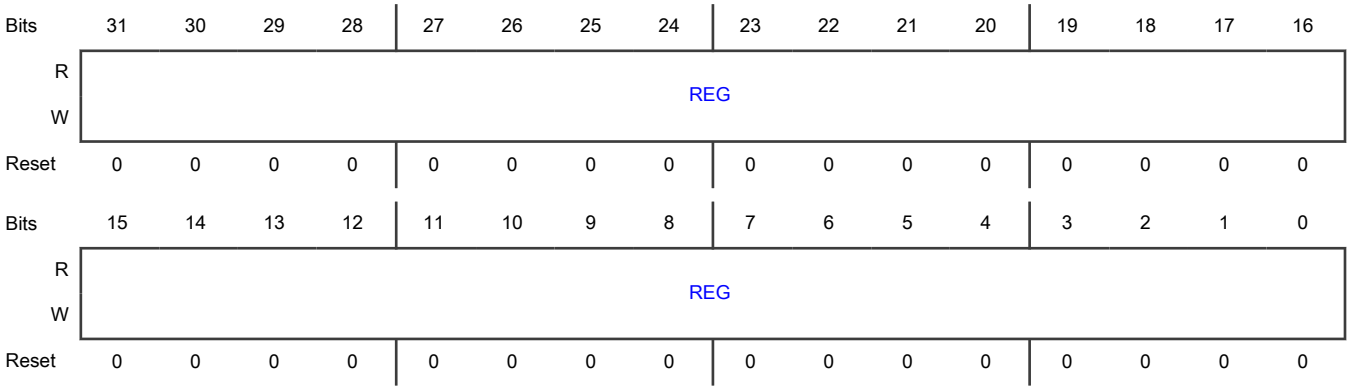
Offset

Register	Offset
WAKEUP0A	700h
WAKEUP1A	708h

Function

Contains software writable bits. Writes to this register are blocked when WAKEUP lock register is set.

Diagram



Fields

Field	Function
31-0 REG	Register Software writable value.

36.7.1.30 Wakeup 0 Register B (WAKEUP0B - WAKEUP1B)

Offset

Register	Offset
WAKEUP0B	704h
WAKEUP1B	70Ch

Function

Contains the inverted Wakeup 0 software writable bits. When configuring the wakeup registers, both wakeup A and wakeup B registers need to be programmed to the appropriate values. Writes to this register are blocked when WAKEUP lock register is set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INVERSE															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INVERSE															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-0 INVERSE	Inverse value Must be programmed with the inverted contents of WAKEUP0A.

36.7.1.31 Wakeup Lock A (WAKLCKA)

Offset

Register	Offset
WAKLCKA	7F8h

Function

Contains the lock bits. Writes to this register are blocked when Wakeup lock registers are set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 LOCK	Lock When set, blocks writes to the Wakeup registers. 0b - Lock is disabled 1b - Lock is enabled

36.7.1.32 Wakeup Lock B (WAKLCKB)

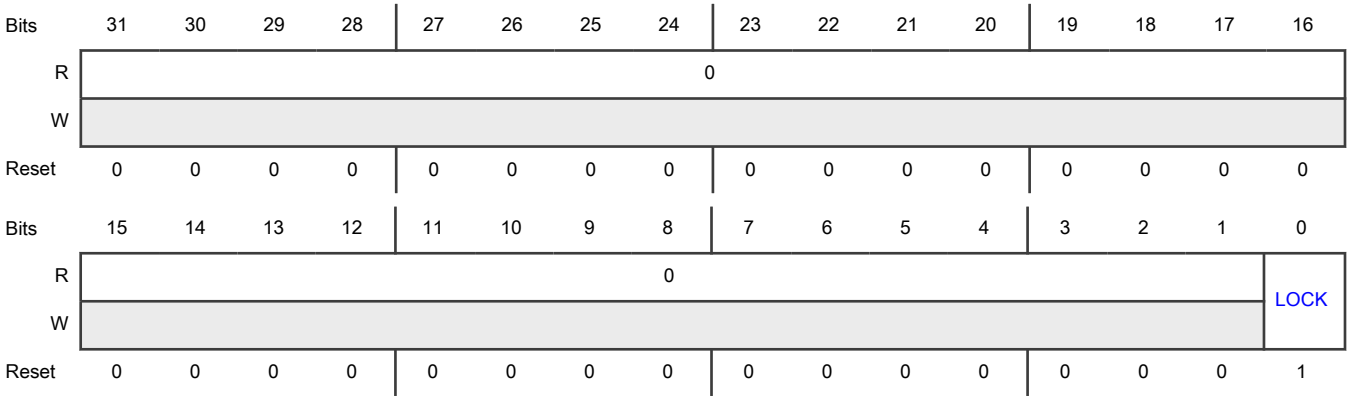
Offset

Register	Offset
WAKLCKB	7FCh

Function

Contains the inverted wakeup lock bits. When configuring the wakeup registers, both lock A and lock B registers need to be programmed to the appropriate values. Writes to this register are blocked when wakeup lock registers are set.

Diagram



Fields

Field	Function
31-1 —	Reserved
0	Lock

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOCK	When clear, blocks writes to the wakeup registers and asserts a configuration error if any of the wakeup A registers are not equal to inverted wakeup B registers. 0b - Lock is enabled 1b - Lock is disabled

Chapter 37
System Power Control (SPC)

37.1 Chip-specific SPC information

Table 341. Reference links to related information

Topic	Related module	Reference
Full description	SPC	SPC
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

The device has analog voltage glitch detector (aGDET) and two digital voltage glitch detectors (GDET) which enhance protection against voltage manipulation. The NXP boot ROM configures aGDET and GDET during device initialization to reset the device when voltage manipulation is detected. The aGDET and GDET were validated on silicon across multiple process, voltage, and temperature corners with maximum noise profile generated by activating a large number of possible boot paths. To avoid false triggering due to unforeseen environment or application behavior, NXP recommends disabling aGDET and GDET in early start-up code. aGDET and/or GDET can be used by the application when security-critical operations are taking place, e.g., cryptographic operation using private or symmetric keys (digital signature, encryption, decryption, etc.), or sensitive decision-making process. However, characterization of the system noise profile and interactions with the sensors are required.

37.1.1 Module instances

This device has one instance of the SPC module, SPC0.

37.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

37.1.3 Voltage regulators

This section describes the various on-chip regulators.

Table 342. On-chip regulators

Regulator name	Description
DCDC_CORE	The DCDC_CORE can be used to power the VDD_CORE domain.
LDO_CORE	The LDO_CORE can be used to power VDD_CORE domain.
LDO_SYS	The LDO_SYS can be used to power VDD_SYS domain. It can power the IO ring, other on-chip modules, and external components as well. However, user should take care that the total load current doesn't exceed the maximum capacity.

37.1.4 Registers for power domains

The CMC module controls two power domains on this device. Below are details how SPC registers map to these power domains:

- PD_STATUS0 refers to power domain VDD_CORE_MAIN.
- PD_STATUS1 refers to power domain VDD_CORE_WAKE.

37.1.5 SPC_SC[ISO_CLR]

SPC_SC[ISO_CLR] controls and gives status of the power domain latching. This field maps to each power domain as follows:

- Bit 16 clears the isolation for VDD_CORE_MAIN domain modules, particularly GPIO/PORTs 1 through 4, DAC0/1 and VREF0.
- Bit 17 clears the isolation for VDD_CORE_WAKE domain modules, particularly GPIO/PORT 0.

37.1.6 SOC_CNTRL fields

ACTIVE_CFG1[SOC_CNTRL] field controls which analog modules are enabled and disabled in Active or Sleep modes. LP_CFG1[SOC_CNTRL] field controls which analog modules are enabled and disabled in other low-power modes (like Deep Sleep, PowerDown, or Deep PowerDown). The following table lists the analog modules that these fields control.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W										CMP2_DAC_EN	CMP1_DAC_EN	CMP0_DAC_EN		CMP2_EN	CMP1_EN	CMP0_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W						OPAMP2_EN	OPAMP1_EN	OPAMP0_EN		DAC2_EN	DAC1_EN	DAC0_EN			USB3v_DET_EN	VREF_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

37.1.7 Low-power current reference enable

The low-power current reference is used by FRO12M and analog modules CMP0/1/2. You must write 1 to LP_CFG[LP_IREFEN] if you use these analog modules in Deep Power Down mode. For the comparators CMP, you must write 1 to this field when using nano mode in Deep Power Down mode.

37.1.8 SPC output signals bits

The following SPC signals are internal to the MCU and can be trigger sources to other peripherals through the INPUTMUX:

- DCDC_BURST_ACTIVE: Indicates if DCDC is actively bursting or in quiet period. It can trigger the CTimers.
- DCDC_BURST_TRIG_PULSE: Pulses when burst ends and quiet period starts. It can trigger ADC.

37.1.9 HVD and LVD for I/O supply rails

The HVD and LVD circuits for the I/O monitor supply rail VDD for PORT0 and PORT1, and the following fields control these circuits:

- ACTIVE_CFG[IO_HVDE]
- ACTIVE_CFG[IO_LVDE]
- LP_CFG[IO_HVDE]
- LP_CFG[IO_LVDE]
- VD_IO_CFG[HVDIE]
- VD_IO_CFG[LVDIE]

You can read the status of these circuits using VD_STAT[IOVDD_HVDF] and VD_STAT[IOVDD_LVDF].

37.1.10 DCDC drive strength

For the MCU low-power modes, active or sleep mode can use the DCDC in normal or low drive strength using ACTIVE_CFG[DCDC_VDD_DS]. In deep sleep or power down mode, the DCDC can be used in normal, low, or pulse refresh drive strength using LP_CFG[DCDC_VDD_DS]. In deep power down mode, the DCDC is powered down.

NOTE

When the chip is in low-power mode and DCDC is during active mode, the DCDC voltage level must be set the same in ACTIVE_CFG and LP_CFG.

37.1.11 DCDC refresh counter clock

The DCDC pulse refresh counter (PULSE_REFRESH_CNT) is clocked by lp_osc.

37.1.12 Bandgap mode

You must configure the bandgap to be enabled if any of the following modules are enabled in active mode (configure ACTIVE_CFG[BGMODE]) or if they are configured to remain enabled in low power modes (configure LP_CFG[BGMODE]).

- FRO_HF (FRO_144M)
- PLL0 or PLL1 or System Oscillator
- CMP (if enabled in high power or normal power mode)
- Any LVD or HVD or Glitch Detect monitor in SPC
- LDO_CORE (if configured for normal drive strength)
- LDO_SYS (if configured for normal drive strength)
- DCDC (if configured for normal drive strength)

37.1.13 LPWKUP_DELAY register configuration

When a different VDD_CORE level is configured in low power modes (LP_CFG register) compared to active modes (ACTIVE_CFG register), the LPWKUP_DELAY register must be configured to a minimum value, as documented in the following table.

Table 343. LPWKUP_DELAY configuration

Regulator	LP_CFG voltage	ACTIVE_CFG voltage	Minimum delay[1]	Minimum LPWKUP_DELAY
LDO_CORE	1.0 V	1.1 V	17 µs	0x00AA
LDO_CORE	1.0 V	1.1 V	31 µs	0x0136
LDO_CORE	1.1 V	1.2 V	17 µs	0x00AA

Table continues on the next page...

Table 343. LPWKUP_DELAY configuration (continued)

Regulator	LP_CFG voltage	ACTIVE_CFG voltage	Minimum delay[1]	Minimum LPWKUP_DELAY
DCDC	0.7 V	1.0 V	95 µs	0x03BB
DCDC	0.7 V	1.1 V	127 µs	0x04F9
DCDC	0.7 V	1.2 V	159 µs	0x0637
DCDC	1.0 V	1.1 V	39 µs	0x0186
DCDC	1.0 V	1.2 V	78 µs	0x030C
DCDC	1.1 V	1.2 V	39 µs	0x0186

1. Delay numbers are calculated assuming the maximum value of external output capacitor is implemented. The minimum delay can be halved if the maximum external output capacitance is also halved.

When the SPC_LPREQ pin is used during low power modes to control an external switch or external PMIC, the LPWKUP_DELAY register must also be used to compensate for any additional wakeup time required by the external device. In this case, the larger of the two values should be used.

37.2 Overview

SPC contains and controls the following components:

- Two low-drop-out (LDO) voltage regulators (1.1 V LDO_CORE and 1.8 V LDO_SYS)
- One DCDC used for regulating system and core power
- Circuits that monitor and assure correct voltage levels:
  - Power-on reset (POR)
  - Low-voltage detect (LVD)
  - Analog glitch detector
  - High-voltage detect (HVD)

SPC turns these components on and off, and switches them to different power mode levels based on power-mode change requests from each of the chip power domains.

37.2.1 Block diagram

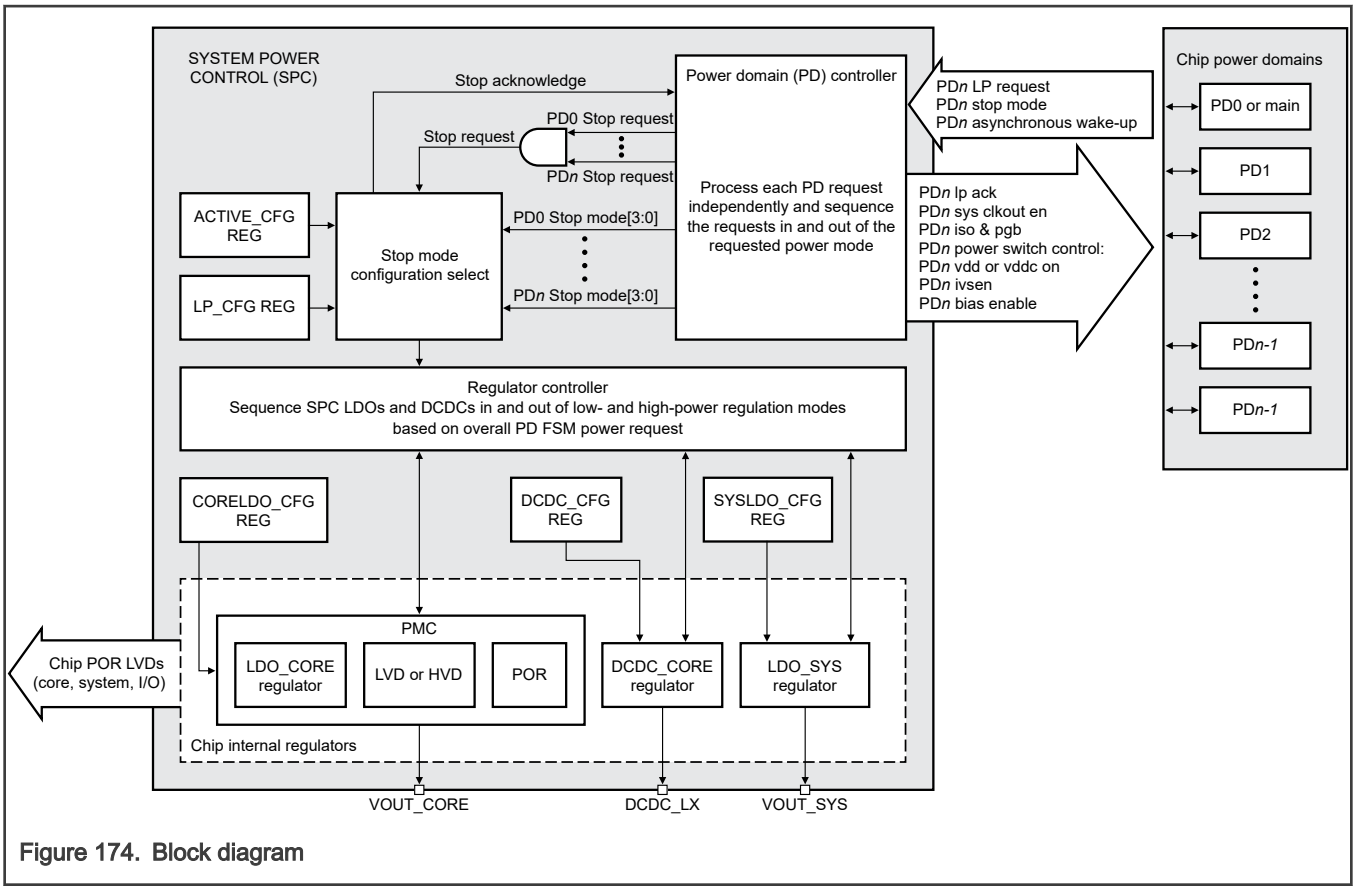


Figure 174. Block diagram

37.2.2 Features

- Selectable core internal voltage regulator (1.1 V LDO_CORE or 1.1 V DCDC)
- Selectable system internal voltage regulator (1.8 V LDO_SYS)
- Active POR providing brown-out detect
- Low-voltage detect
- High-voltage detect
- Core power gate control
- Analog glitch detector

37.3 Functional description

37.3.1 Power mode transitions

Figure 175 shows the power mode state transitions available for each power-down domain. A POR or LVD reset initially returns the power-down domains to ACTIVE state.

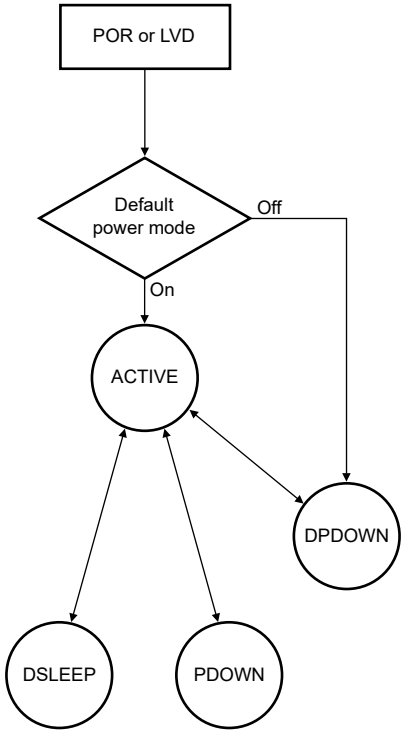


Figure 175. Power mode state diagram

SPC allows each power domain to enter or exit low-power modes independently of the other. You must ensure that any common system resources (for example, clock sources) are always available whenever required for a power-domain current power mode. For example, if a clock source in power domain 0 drives power domain 1, you must ensure that power domain 0 does not enter a low-power mode, because that would disable the clock source.

Figure 176 defines the low-power entry and exit flow for each of the SPC power-down domains.

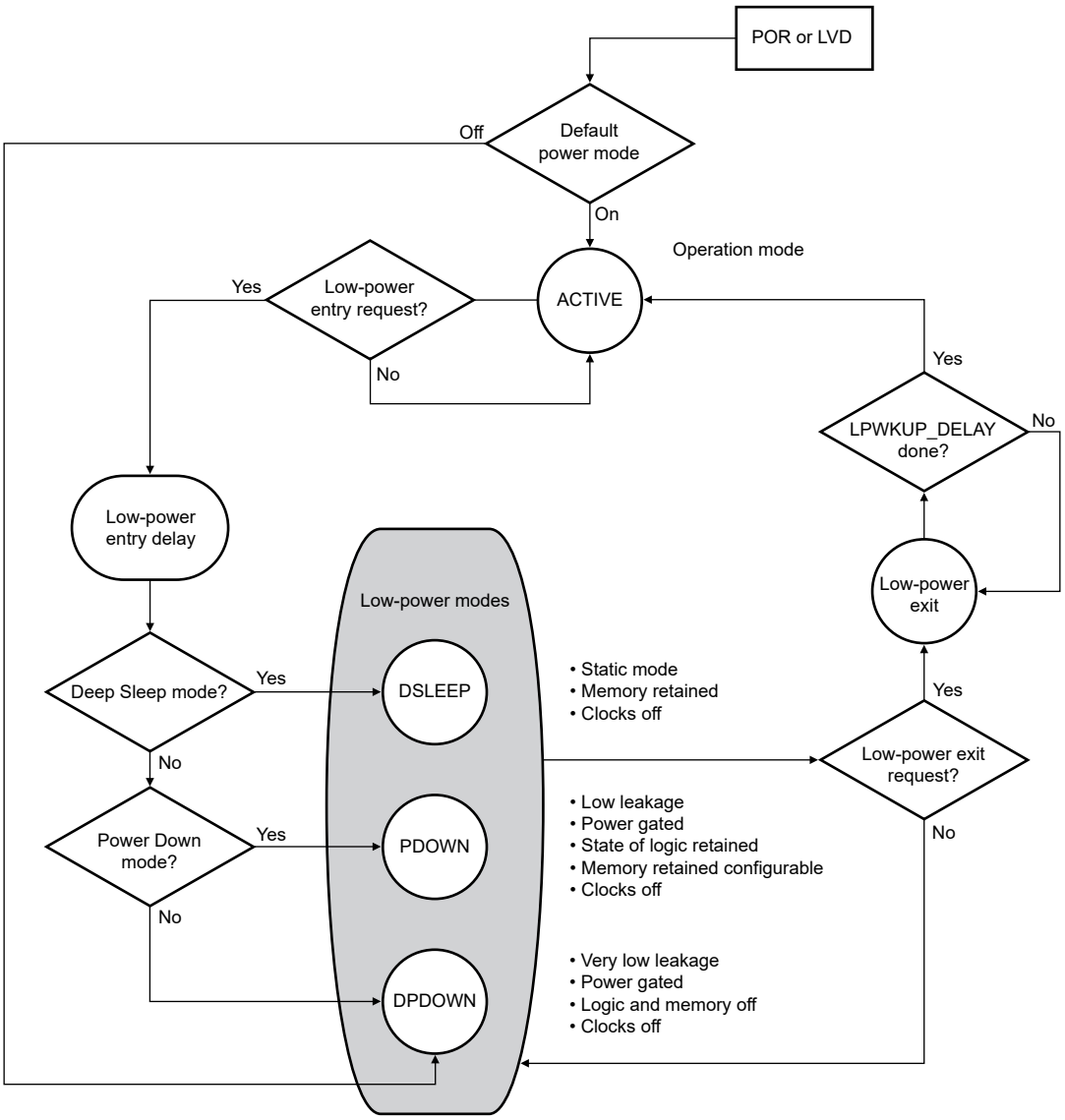


Figure 176. SPC regulator power mode state diagram

NOTE

This section mentions the power modes supported by the module. For the power modes supported on your device, see the chip-specific Power management chapter.

37.3.2 Active mode voltage updates

This section describes voltage and frequency changes in Active mode.

You can alter the operating voltage and frequency during Active mode to optimize power consumption depending on the application's performance requirements.

When increasing voltage and frequency in Active mode, you must perform the following steps:

1. Increase voltage to a new level (ACTIVE_CFG[CORELDO_VDD_LVL] and/or ACTIVE_CFG[DCDC_VDD_LVL]).
2. Wait for voltage change to complete (SC[BUSY] = 0).
3. Configure flash memory to support higher voltage level and frequency (MSF_FCTRL[RWSC]).

4. Configure SRAM to support higher voltage levels (SRAMCTL[VSM]).
5. Request SRAM voltage update (write 1 to SRAMCTL[REQ]).
6. Wait for SRAM voltage change to complete (SRAMCTL[ACK] = 1).
7. Clear request for SRAM voltage change (write 0 to SRAMCTL[REQ]).
8. Increase frequency to a new level (for example, SCG_RCCR).
9. You can continue execution.

When decreasing voltage and frequency in Active mode perform the following steps:

1. Decrease frequency to a new level (for example, SCG_RCCR).
2. Configure flash memory to support lower voltage level and frequency (MSF_FCTRL[RWSC]).
3. Configure SRAM to support lower voltage levels (SRAMCTL[VSM]).
4. Request SRAM voltage update (write 1 to SRAMCTL[REQ]).
5. Wait for SRAM voltage change to complete (SRAMCTL[ACK] = 1).
6. Clear request for SRAM voltage change (write 0 to SRAMCTL[REQ]).
7. Decrease voltage to a new level (ACTIVE_CFG[CORELDO_VDD_LVL] and/or ACTIVE_CFG[DCDC_VDD_LVL]).
8. Wait for voltage change to complete (SC[BUSY] = 0).
9. You can continue execution.

37.3.3 Low-Power Request (LPREQ) pin

The LPREQ pin asserts after low-power entry and negates after low-power wakeup.

The pin is intended to communicate with an external PMIC (for example, to enter low-power mode) or to control external switches (for example, if certain power supply rails are switched off in the low-power mode). The pin can eliminate the latency when communicating with PMIC—for example, through I^2C—for quick power-state transitions. The PMIC can be configured to use the LPREQ pin to switch between two preconfigured power states.

SPC controls the state of the LPREQ pin based on how you configure Low-Power Request Configuration (LPREQ_CFG). You control the LPREQ pin in Active mode. SPC controls the pin when the chip transitions from Active to a low-power mode, and after wake-up from these power modes.

NOTE

If the power supply rails are switched off externally, software must configure the internal isolation of these power domains using the External Voltage Domain Configuration (EVD_CFG) register.

To use the LPREQ pin:

1. Specify the pin polarity (LPREQ_CFG[LPREQPOL]).
2. Enable the pin output (LPREQ_CFG[LPREQOE]).
3. Configure the pin mux for the desired pin using the PORT PCR registers.
4. If the external PMIC or switch needs additional time after wake-up, use Low Power Wake-Up Delay (LPWKUP_DELAY) to extend the wake-up time.

37.3.4 DCDC

DCDC is a hysteretic converter. It is active during a burst period, when it switches the output to charge to a specific voltage. Then DCDC stops switching for a quiet period until the output voltage drops to a lower level, and it does another active burst.

37.3.4.1 DCDC drive strength

DCDC supports multiple drive strength options that you can choose:

- Normal
- Low
- Pulse refresh

The normal and low drive strengths differ in these DCDC characteristics:

- Maximum load current
- Quiescent current
- Transient response

With low drive strength, you can benefit from the lower quiescent current if:

- The load current is below the I_{LOAD} specification in the data sheet.
- Your application can tolerate the slower transient response.

Be careful when choosing low drive strength. If you exceed I_{LOAD} or require faster transient response, you must use normal drive strength. Furthermore, most other DCDC features are only available with normal drive strength.

Pulse-refresh drive strength offers even lower quiescent current in low-power modes, by deactivating DCDC when not active. This strength has an active pulsing period where DCDC charges the output voltage. Then DCDC deactivates for a configurable amount of time. No analog comparator is powered to monitor the DCDC output voltage. Instead, this drive strength is time-based using an SPC timer where DCDC_BURST_CFG[PULSE_REFRESH_CNT] specifies the timer period. Every period, SPC enables DCDC to charge the voltage again. When DCDC is off, no voltage regulation occurs. You must specify DCDC_BURST_CFG[PULSE_REFRESH_CNT] to manage the voltage drop on the DCDC capacitor and avoid a brownout condition.

37.3.4.2 DCDC burst

You can control and monitor the DCDC burst. Synchronizing with the quiet period between bursts can be valuable for noise-sensitive applications—for example, during a high-resolution ADC measurement or a radio transmission.

Before requesting a burst to start, clear the DCDC_BURST_CFG[BURST_ACK] flag. Then, your firmware can request a burst by writing 1 to DCDC_BURST_CFG[BURST_REQ]. SPC sets the DCDC_BURST_CFG[BURST_ACK] flag when the burst has completed and DCDC enters a quiet period.

SPC also provides signals to synchronize the burst activity with other peripherals in the chip. If DCDC_BURST_CFG[EXT_BURST_EN] = 1 before a burst starts, the output DCDC_BURST_TRIG_PULSE signal asserts when a burst completes, and can trigger other actions such as an ADC conversion. Another SPC output signal is DCDC_BURST_ACTIVE, which EXT_BURST_EN does not gate. DCDC_BURST_ACTIVE can also trigger other peripherals such as a timer. You can use this signal to estimate the power that DCDC delivers.

When bursting, you can control the DCDC switching center frequency with the frequency-stabilization feature. DCDC_CFG[FREQ_CNTRL_ON] enables this feature, and when the feature is enabled, DCDC_CFG[FREQ_CNTRL] controls it. The main parameters that control the period between active bursts and quiet period are:

- Load current
- External capacitor
- DCDC hysteresis

NOTE

Writing 1 to DCDC_CFG[FREQ_CNTRL_ON] stabilizes the frequency by limiting DCDC current. See the DCDC I_{LOAD} maximum limit in the data sheet for this mode.

37.3.4.3 DCDC voltage changes

To support power optimization of applications, DCDC can provide multiple output voltages.

Two separate registers, which specify Active mode output and Low-Power mode output, independently control the DCDC output voltage. Whenever an SoC mode change occurs, by using two independent registers, there is no need to reconfigure the output voltage. ACTIVE_CFG[DCDC_VDD_LVL] controls the DCDC output voltage when the SoC mode is Active or in Sleep mode. The DCDC drive strength is normal (ACTIVE_CFG[DCDC_VDD_DS] = 10b). LP_CFG[DCDC_VDD_LVL] controls the DCDC output voltage when the SoC mode is in a Low-Power mode (i.e., not Active or Sleep mode) or the DCDC drive strength is low (ACTIVE_CFG[DCDC_VDD_DS] = 01b) and the SoC is in an Active or Sleep mode.

When Active Power Mode Configuration (ACTIVE_CFG) controls the output, following are the DCDC output options:

- 1.0 V
- 1.1 V
- 1.2 V

Align the output voltage with the desired clock frequencies of the system. When changing the DCDC output voltage level, take care to change the CORE LDO voltage level (ACTIVE_CFG[CORELDO_VDD_LVL]) to the same level (even if you turn off the CORE LDO). ACTIVE_CFG[CORELDO_VDD_LVL] also controls the voltage threshold of the LVD monitoring the VDD_CORE supply. You must adjust it with the DCDC output voltage.

See Active mode voltage updates to change the DCDC output level in Active mode.

When Low-Power Mode Configuration (LP_CFG) controls the output, following are the DCDC output options:

- 0.7 V
- 1.0 V
- 1.1 V
- 1.2 V

When entering SoC power-down mode with IVS disabled, support only the 0.7 V option. You have to configure LPWKUP_DELAY to the correct value. In Deep Sleep mode, you must also align the output voltage with the clock frequencies of any clocks kept alive in the Low-Power mode. If you keep some clock sources active, ensure to leave the bandgap enabled.

When changing the DCDC output voltage level, you must change the CORE LDO voltage level (LP_CFG[CORELDO_VDD_LVL]) to the same level (even when you turn off the CORE LDO). LP_CFG[CORELDO_VDD_LVL] also controls the voltage threshold of the LVD monitoring the VDD_CORE supply. You must adjust it with the DCDC output voltage.

To change the DCDC output level in Low-Power mode, follow this procedure:

1. Configure LP_CFG[DCDC_VDD_LVL] to desired level.
2. Configure LP_CFG[DCDC_VDD_DS] = 0x1.
3. Configure ACTIVE_CFG[DCDC_VDD_LVL] to same level as in configure LP_CFG[DCDC_VDD_LVL].

37.3.5 Clocks

Table 344. Clocks

SPC component:	Clock source
Registers and logic	Slow bus clock
Delay counters	10 MHz internal clock

37.3.6 Reset

37.3.6.1 Reset sources

37.3.6.1.1 Power-on reset (POR)

When you initially apply power to the chip, or the supply voltage is below the POR falling threshold, the POR circuit triggers the POR condition.

The POR condition asserts a cold reset in all power domains. The SRS[POR] and SRS[LVD] status bits are both set after a POR condition.

37.3.6.1.2 Low-voltage detect (LVD)

SPC supports the following LVD circuits:

- Core VDD
- System VDD
- I/O VDD

Each of these circuits is enabled by default and keeps the chip in reset until each of these supply voltages rises above the LVD rising threshold. Enabling any of the LVD circuits triggers an LVD reset condition if the corresponding supply voltage is below the LVD falling threshold.

The LVD reset condition asserts a cold reset in all power domains. The LVD- and HVD-detection logic is the only logic that HVD and LVD resets do not affect. The corresponding status fields in Voltage Detect Status (VD_STAT) become 1 after the corresponding LVD reset condition occurs.

37.3.6.1.3 High-voltage detect (HVD)

SPC supports the following HVD circuits:

- Core VDD
- System VDD
- I/O VDD

Each of these circuits is enabled by default and triggers an HVD reset condition if the corresponding supply voltage is above the HVD threshold.

Any of these reset condition asserts a cold reset in all power domains. The LVD- and HVD-detection logic is the only logic that HVD and LVD resets do not affect. The corresponding status fields in Voltage Detect Status (VD_STAT) become 1 after the corresponding HVD reset condition occurs.

37.3.6.1.4 Glitch detect

This chip guards against short pulses on the core VDD and VSS. This security feature can detect glitches due to a hacking attack and alerts the core to reset the chip or generate a interrupt operation.

After power-on-reset, the core VDD glitch-detect system is enabled. When the system is in Active or Sleep mode, use ACTIVE_CFG[GLITCH_DETECT_DISABLE] to enable or disable the glitch-detect system. When the system is in other Low-Power mode, use LP_CFG[GLITCH_DETECT_DISABLE] to enable or disable the glitch-detect system.

The glitch-detect system uses a 4-bit ripple counter to monitor the quantity of detected glitches. The VDD_CORE_GLITCH_DETECT_SC[GLITCH_DETECT_FLAG] status flags capture the state of each of these counters.

37.3.6.2 Reset sequence

The following steps occur as part of the POR, LVD, or HVD sequence:

1. The RESET_b pin deasserts.
2. The internal reset signals assert.
3. The POR or LVD signals negate.

4. System clocks are enabled.

37.3.7 Interrupts

SPC generates a single interrupt. The trigger for this interrupt comes from:

- Any of the LVD, HVD, or glitch-detect circuits when they are not configured for reset and their corresponding HVDIE or LVDIE register.
- The Interrupt Enable (IE) register (for the glitch-detect case).

37.4 External signals

Signal	Description	Direction
LPREQ	Low-power request pin used to signal external power-management circuits for a change in supply voltage	Output
VOUT_CORE	Output of the LDO_CORE regulator	Output
VOUT_SYS	Output of the LDO_SYS regulator	Output
DCDC_LX	Output of the DCDC_CORE regulator	Output

37.5 Initialization

To initialize SPC with LDO_CORE off, write 0 to CNTRL[CORELDO_EN].

To initialize SPC with LDO_SYS off, write 0 to CNTRL[SYSLDO_EN].

To initialize SPC with DCDC_CORE off, write 0 to CNTRL[DCDC_EN].

Except for the above, SPC does not require any special initialization.

37.6 Application information

To change DCDC level in Low-Power mode:

1. Configure LP_CFG[DCDC_VDD_LVL] to desired level.
2. Configure LP_CFG[DCDC_VDD_DS] = 0x1
3. Configure ACTIVE_CFG[DCDC_VDD_LVL] to same level programmed in Configure LP_CFG[DCDC_VDD_LVL]

To disable Bandgap in Active or Sleep mode:

1. Disable all LVD's and HVD's in ACTIVE_CFG[29:24]= 0x00
2. Disable Glitch Detect, ACTIVE_CFG[GLITCH_DETECT_DISABLE] = 1
3. Configure LDO's and DCDC to Low Drive Strength in ACTIVE_CFG register
4. Configure ACTIVE_CFG[BGMODE] = 0x0

To disable Bandgap in Low-Power mode:

1. Disable all LVD's and HVD's in LP_CFG[29:24]= 0x00
2. Disable Glitch Detect, LP_CFG[GLITCH_DETECT_DISABLE] = 1
3. Configure LDO's and DCDC to Low Drive Strength in LP_CFG register
4. Configure LP_CFG[BGMODE] = 0x0

37.7 SPC register descriptions

Different chip reset types affect the reset of different SPC registers. Each register description provides details. See the Reset chapter for more information about the types of reset on this chip.

You may use only 32-bit writes for any writable SPC registers. 8-bit or 16-bit writes cause a transfer error.

37.7.1 SPC memory map

SPC0 base address: 4004_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0000_0000h
10h	Status Control (SC)	32	RW	0000_0000h
14h	SPC Regulator Control (CNTRL)	32	RW	0000_0007h
1Ch	Low-Power Request Configuration (LPREQ_CFG)	32	RW	0000_0000h
30h - 34h	SPC Power Domain Mode Status (PD_STATUS0 - PD_STATUS1)	32	RW	0000_0000h
40h	SRAM Control (SRAMCTL)	32	RW	0000_0001h
100h	Active Power Mode Configuration (ACTIVE_CFG)	32	RW	3F10_0615h
104h	Active Power Mode Configuration 1 (ACTIVE_CFG1)	32	RW	0000_0002h
108h	Low-Power Mode Configuration (LP_CFG)	32	RW	0002_1504h
10Ch	Low Power Mode Configuration 1 (LP_CFG1)	32	RW	0000_0002h
120h	Low Power Wake-Up Delay (LPWKUP_DELAY)	32	RW	0000_0000h
124h	Active Voltage Trim Delay (ACTIVE_VDELAY)	32	RW	0000_00C8h
130h	Voltage Detect Status (VD_STAT)	32	RW	0000_0000h
134h	Core Voltage Detect Configuration (VD_CORE_CFG)	32	RW	0000_0001h
138h	System Voltage Detect Configuration (VD_SYS_CFG)	32	RW	0000_0001h
13Ch	IO Voltage Detect Configuration (VD_IO_CFG)	32	RW	0000_0101h
140h	External Voltage Domain Configuration (EVD_CFG)	32	RW	0000_0000h
144h	Glitch Detect Status Control (GLITCH_DETECT_SC)	32	RW	0000_003Fh
300h	LDO_CORE Configuration (CORELDO_CFG)	32	RW	0000_0000h
400h	LDO_SYS Configuration (SYSLDO_CFG)	32	RW	0000_0101h
500h	DCDC Configuration (DCDC_CFG)	32	RW	0000_0000h
504h	DCDC Burst Configuration (DCDC_BURST_CFG)	32	RW	0140_0000h

37.7.2 Version ID (VERID)

Offset

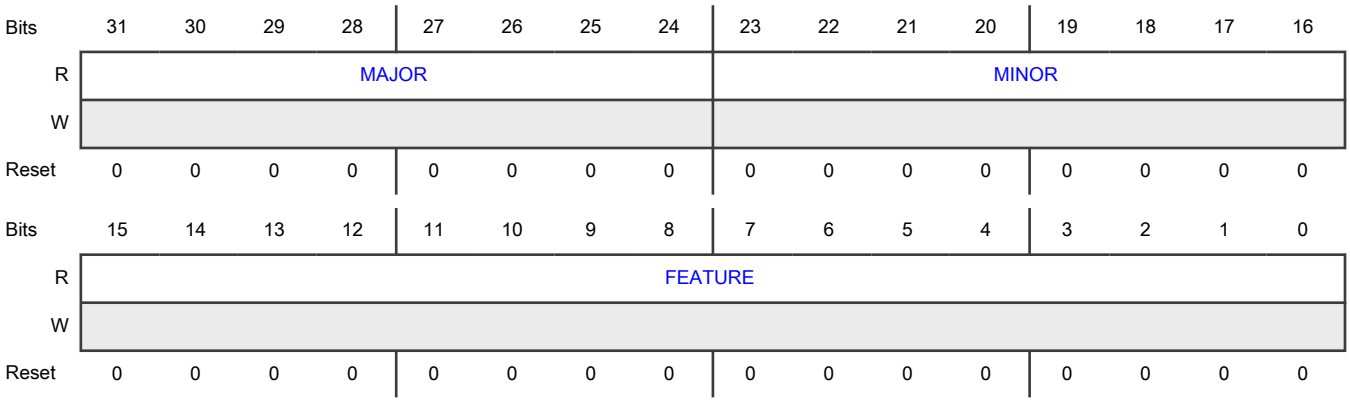
Register	Offset
VERID	0h

Function

Indicates:

- The version integrated for this instance on the chip.
- Inclusion or exclusion of several optional features.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number
23-16 MINOR	Minor Version Number
15-0 FEATURE	Feature Specification Number Indicates the feature set number. 0000_0000_0000_0000b - Standard features All other values are reserved.

37.7.3 Status Control (SC)

Offset

Register	Offset
SC	10h

Function

Indicates the current SPC status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0													ISO_CLR	
W															W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SPC_LP_MODE				0	0	SPC_LP_...	BUSY
W															W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-18 —	Reserved
17-16 ISO_CLR	Isolation Clear Flags Indicates whether certain peripherals and I/O pads are in a latched state as a result of having been in PDOWN mode. Each bit of this field is a flag associated with a module on this chip. See the chip-specific SPC information for the module-bit association. For each flag: • A value of 0 means that peripherals and I/O pads are in the normal run state. • A value of 1 means that peripherals and I/O pads can retain their state in their power domain. If a flag in this field is 1, writing 1 to that flag clears the flag and releases the I/O pads and peripherals to their normal run-mode state.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	After recovering from a power-down mode, you must restore the chip configuration before clearing a flag in this field. In particular, you must restore the pin configuration for enabled WUU wake-up to avoid any WUU flag from being falsely set when the associated ISO_CLR flag is cleared. This field resets after a system reset.
15-8 —	Reserved
7-4 SPC_LP_MODE	Power Domain Low-Power Mode Request Indicates the last low-power mode that the power domain requested. 0000b - Sleep mode with system clock running 0001b - DSLEEP with system clock off 0010b - PDOWN with system clock off 0100b - Reserved 1000b - DPDOWN with system clock off
3 —	Reserved
2 —	Reserved
1 SPC_LP_REQ	SPC Power Mode Configuration Status Flag Indicates when all power-down domains requested low-power mode and SPC entered a low-power state. **NOTE** This field behaves differently for register reads and writes. When reading 0b - SPC is in Active or Sleep mode; the ACTIVE_CFG register has control 1b - All power domains requested low-power mode; SPC entered a low-power state; power-mode configuration based on the LP_CFG register When writing 0b - No effect 1b - Clear the flag
0 BUSY	SPC Busy Status Flag Indicates whether SPC is busy. SPC sets this flag:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• When SPC executes any type of power mode transition in Active mode or any of the chip low-power modes. • Due to LP_CFG[CORELDO_VDD_LVL] and LP_CFG[DCDC_VDD_LVL] changes while in Active mode. **NOTE** Wait until this flag is clear before changing power-mode configuration registers. 0b - Not busy 1b - Busy

37.7.4 SPC Regulator Control (CNTRL)

Offset

Register	Offset
CNTRL	14h

Function

Enables the SPC regulators.

You can write to the fields in this register only one time.

The register resets only after a POR, LVD, or HVD event.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													DCDC_EN	SYSLDO_...	CORELDO...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

Field	Function
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2 DCDC_EN	DCDC_CORE Regulator Enable 0b - Disable 1b - Enable
1 SYSLDO_EN	LDO_SYS Regulator Enable 0b - Disable 1b - Enable
0 CORELDO_EN	LDO_CORE Regulator Enable 0b - Disable 1b - Enable

37.7.5 Low-Power Request Configuration (LPREQ_CFG)

Offset

Register	Offset
LPREQ_CFG	1Ch

Function

Configures the low-power output request pin.

The register resets only after a POR, LVD, or HVD event.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												LPREQOV		LPREQPOL	LPREQOE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-4 —	Reserved
3-2 LPREQOV	Low-Power Request Output Override Forces the low-power request pin high. 00b - Not forced 01b - Reserved 10b - Forced low (ignore LPREQPOL settings) 11b - Forced high (ignore LPREQPOL settings)
1 LPREQPOL	Low-Power Request Output Pin Polarity Control Controls the true polarity of the low-power request output pin. 0b - High 1b - Low
0 LPREQOE	Low-Power Request Output Enable Enables the low-power request output pin. 0b - Disable 1b - Enable

37.7.6 SPC Power Domain Mode Status (PD_STATUS0 - PD_STATUS1)

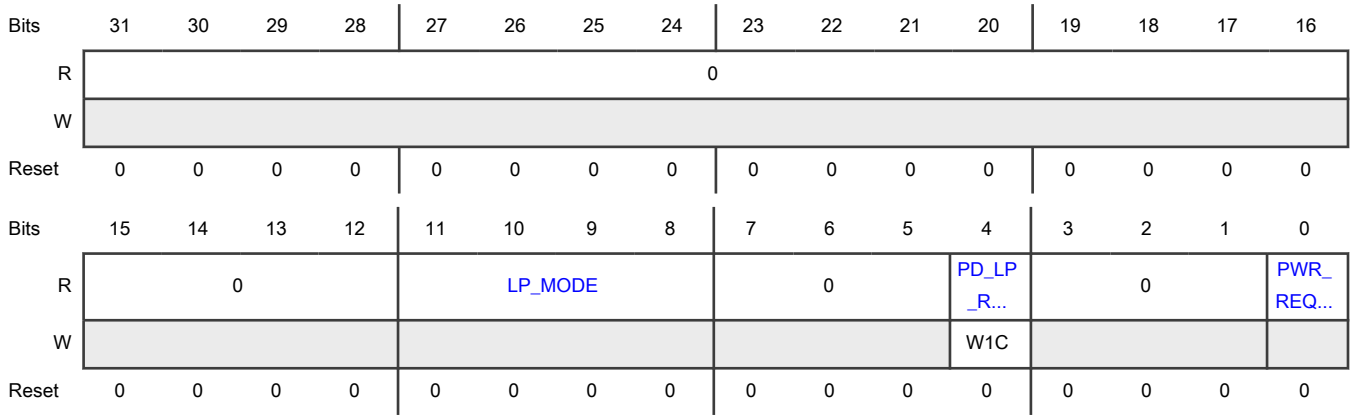
Offset

Register	Offset
PD_STATUS0	30h
PD_STATUS1	34h

Function

Indicates power mode status for each of the SPC power domains. See the chip-specific SPC information to determine which power domains are associated with this register.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-8 LP_MODE	Power Domain Low Power Mode Request Indicates the last low-power mode that the power domain requested. 0000b - SLEEP with system clock running 0001b - DSLEEP with system clock off 0010b - PDOWN with system clock off 0100b - Reserved 1000b - DPDOWN with system clock off
7-5 —	Reserved
4 PD_LP_REQ	Power Domain Low Power Request Flag Set when low power mode was requested by Power Domain since last cleared by software. 0b - Did not request 1b - Requested
3-1 —	Reserved
0 PWR_REQ_STATUS	Power Request Status Flag Indicates whether the power domain requested low-power mode. 0b - Did not request 1b - Requested

37.7.7 SRAM Control (SRAMCTL)

Offset

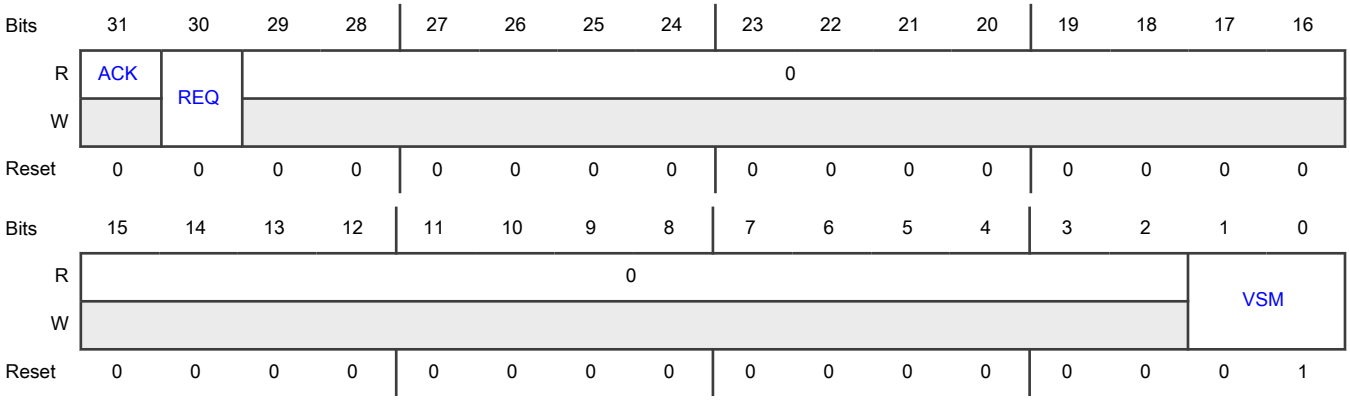
Register	Offset
SRAMCTL	40h

Function

Configures the SRAM timing for different voltage levels.

When transitioning between two voltage levels, you must keep the SRAM timing at the lowest voltage level until the voltage change is complete.

Diagram



Fields

Field	Function
31 ACK	SRAM Voltage Update Request Acknowledge Indicates whether SPC acknowledged the request for an SRAM trim-value change. 0b - Not acknowledged 1b - Acknowledged
30 REQ	SRAM Voltage Update Request Allows you to request an SRAM trim value change. After requesting the change, you must wait for the ACK field to become 1, then write 0 to REQ. 0b - Do not request 1b - Request
29-2 —	Reserved
1-0	Voltage Select Margin

Table continues on the next page...

Table continued from the previous page...

Field	Function
VSM	Specifies the operating voltage for the SRAM's read/write timing margin. 00b - Reserved 01b - 1.0 V 10b - 1.1 V 11b - Reserved

37.7.8 Active Power Mode Configuration (ACTIVE_CFG)

Offset

Register	Offset
ACTIVE_CFG	100h

Function

Controls the SPC regulators settings in the Active and Sleep power mode.

Changes to the drive strength or voltage level for any of the LDOs or DCDC sets the SC[BUSY] flag until SPC changes its state to the new value.

NOTE

The LVDE and HVDE fields reset only with a POR. All other fields reset only with a system reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		IO_HVDE	SYS_HVDE	CORE_HV...	IO_LVDE	SYS_LVDE	CORE_LV...	VDD_VD_...	0	BGMODE		0	LPBUFF_...	0	
W																
Reset	0	0	1	1	1	1	1	1	0	0	0	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			GLITCH_...	DCDC_VDD_LVL		DCDC_VDD_DS		0	SYSLDO_...	0	SYSLDO_...	CORELDO_VDD_LVL		0	CORELDO...
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1

Fields

Field	Function
31-30 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 IO_HVDE	IO High-Voltage Detection Enable Enables the IO high-voltage detection. When IO_HVDE = 1, you must write a value to BGMODE that enables the bandgap. Only a POR resets this field. Monitors VDD IO on PORT. For port details, see the chip-specific information. 0b - Disable 1b - Enable
28 SYS_HVDE	System High-Voltage Detection Enable Enables the system high-voltage detection. When SYS_HVDE = 1, you must write a value to BGMODE that enables the bandgap. Only a POR resets this field. 0b - Disable 1b - Enable
27 CORE_HVDE	Core High-Voltage Detection Enable Enables the core high-voltage detection. When CORE_HVDE = 1, you must write a value to BGMODE that enables the bandgap. Only a POR resets this field. 0b - Disable 1b - Enable
26 IO_LVDE	IO Low-Voltage Detection Enable Enables the IO low-voltage detection. When IO_LVDE = 1, you must write a value to BGMODE that enables the bandgap. Only a POR resets this field. Monitors VDD IO on PORT. For port details, see the chip-specific information. 0b - Disable 1b - Enable
25 SYS_LVDE	System Low-Voltage Detection Enable Enables the system low-voltage detection. When SYS_LVDE = 1, you must write a value to BGMODE that enables the bandgap. Only a POR resets this field. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
24 CORE_LVDE	Core Low-Voltage Detection Enable Enables the core low-voltage detection. When CORE_LVDE = 1, you must write a value to BGMODE that enables the bandgap. Only a POR resets this field. 0b - Disable 1b - Enable
23 VDD_VD_DISABLE	VDD Voltage Detect Disable Disables VDD voltage detect. Mask all LVDs and HVDs when you change the active voltage levels. If this field is 1, it will not mask and allow LVDs and HVDs conditions to generate a system reset or interrupt. If this field is 0, it will mask all LVDs and HVDs when changing the regulator voltage levels during Active mode. 0b - Enable 1b - Disable
22 —	Reserved
21-20 BGMODE	Bandgap Mode Specifies the bandgap mode configuration. BGMODE will be set to 01b to keep the Bandgap enabled if a write to disable Bandgap is detected while keeping any of the regulators in normal drive strength or if any of the LVD/HVDs are kept enabled or if VDD CORE glitch detect are kept enabled. 00b - Bandgap disabled 01b - Bandgap enabled, buffer disabled 10b - Bandgap enabled, buffer enabled 11b - Reserved
19 —	Reserved
18 LPBUFF_EN	CMP Bandgap Buffer Enable Enables the buffer-stored reference voltage to CMP. 0b - Disable 1b - Enable
17-13 —	Reserved
12	Glitch Detect Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
GLITCH_DETECT_DISABLE	Reset on POR only State of GLITCH_DETECT_DISABLE will be ignored if Bandgap is disabled and Glitch Detect hardware will be forced to OFF state. 0b - Low Voltage Glitch Detect enabled 1b - Low Voltage Glitch Detect disabled
11-10 DCDC_VDD_LVL	DCDC VDD Regulator Voltage Level Specifies the DCDC VDD regulator level. When switching the DCDC drive strength from low to normal, ensure that the DCDC high VDD LVL setting is set to the same level that was set prior to switching the DCDC to low drive strength. Otherwise, if the LVDs are enabled, an unexpected LVD can occur. 00b - Reserved 01b - Midvoltage (1.0 V) 10b - Normal voltage (1.1 V) 11b - Overdrive voltage (1.2 V)
9-8 DCDC_VDD_DS	DCDC VDD Drive Strength Specifies the DCDC VDD regulator drive strength. If you specify normal drive strength, you must write a value to ACTIVE_CFG[BGMODE] that enables the bandgap. 01b - Low 10b - Normal All other values are reserved.
7 —	Reserved
6 SYSLDO_VDD_LVL	LDO_SYS VDD Regulator Voltage Level Specifies the LDO_SYS VDD regulator level. You must write 0 to SYS_HVDE before writing 1 to SYSLDO_VDD_LVL. The VDD_SYS voltage can only operate at the overdrive voltage for a limited amount of time for the life of the chip. See the chip data sheet for the maximum voltage and time specifications. This is intended only for programming eFuses. 0b - Normal voltage (1.8 V) 1b - Overdrive voltage (2.5 V)
5 —	Reserved
4	LDO_SYS VDD Drive Strength

Table continues on the next page...

Table continued from the previous page...

Field	Function
SYSLDO_VDD_DS	Specifies the LDO_SYS VDD regulator drive strength If you specify normal drive strength, you must write a value to ACTIVE_CFG[BGMODE] that enables the bandgap. If you have enabled LVDs or HVDs, and attempt to specify low drive strength, SPC ignores the attempt. SPC forces a normal drive strength when SYSLDO_VDD_LVL = 1. 0b - Low 1b - Normal
3-2 CORELDO_VDD_LVL	LDO_CORE VDD Regulator Voltage Level **NOTE** LDO_CORE overdrive voltage level (1.2V) is not supported when LDO_CORE is configured for low drive. Selects the LDO_CORE VDD regulator level. If the CORELDO_VDD_DS fields are set to the same value in both the ACTIVE_CFG and LP_CFG registers, the CORELDO_VDD_LVL's in the ACTIVE_CFG and LP_CFG register must be set to the same voltage level settings. You can change the core VDD levels for the LDO_CORE low power regulator only when CORELDO_VDD_DS=1. When switching CORELDO_VDD_DS from low to normal drive strength, ensure the LDO_CORE high VDD LVL setting is set to the same level that was set prior to switching to the LDO_CORE drive strength (CORELDO_VDD_DS). Otherwise, if the LVDs are enabled, an unexpected LVD can occur. You must specify the same level for both CORELDO_VDD_LVL and DCDC_VDD_LVL, even if LDO_CORE is off. 00b - Reserved 01b - Regulate to mid voltage (1.0 V) 10b - Regulate to normal voltage (1.1 V) 11b - Regulate to overdrive voltage (1.2 V)
1 —	Reserved
0 CORELDO_VDD_DS	LDO_CORE VDD Drive Strength Selects the LDO_CORE VDD regulator drive strength **NOTE** When setting CORELDO_VDD_DS to Normal, BGMODE must be programmed to a value that enables the Bandgap. Writes to set drive strength to Low will be ignored if LVD/HVDs are kept enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Low 1b - Normal

37.7.9 Active Power Mode Configuration 1 (ACTIVE_CFG1)

Offset

Register	Offset
ACTIVE_CFG1	104h

Function

Enables different analog modules in Active or Sleep mode. To see what analog modules each of these bits are controlling, see the chip-specific SPC information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SOC_CNTRL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SOC_CNTRL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-0 SOC_CNTRL	Active Config Chip Control Enables analog modules in Active or Sleep mode. Bit combinations with 0: The associated analog modules are disabled. Bit combinations with 1: The associated analog modules are enabled.

37.7.10 Low-Power Mode Configuration (LP_CFG)

Offset

Register	Offset
LP_CFG	108h

Function

Controls the SPC regulator settings in Deep Sleep, Power Down, and Deep Power Down modes.

This register resets only after a POR or LVD event.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		IO_HVDE	SYS_HVDE	CORE_HV...	IO_LVDE	SYS_LVDE	CORE_LV...	LP_IREF...	0	BGMODE		0	LPBUFF_...	CORE VDD...	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			GLITCH_...	DCDC_VDD_LVL		DCDC_VDD_DS		0	0		SYSLDO_...	CORELDO_VDD_LVL		0	CORE LDO...
W																
Reset	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0

Fields

Field	Function
31-30 —	Reserved
29 IO_HVDE	IO High Voltage Detect Enable This field monitors VDD IO on PORT. For port details, see the chip-specific SPC information. This field resets only after a POR, LVD, or HVD event. When IO_HVDE = 1, you must use LP_CFG[BGMODE] to enable the bandgap. If you enable the bandgap to support voltage detection, the low-power mode Idd increases. 0b - Disable 1b - Enable
28 SYS_HVDE	System High Voltage Detect Enable This field resets only after a POR, LVD, or HVD event. When SYS_HVDE = 1, you must use LP_CFG[BGMODE] to enable the bandgap. If you enable the bandgap to support voltage detection, the low-power mode Idd increases.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
27 CORE_HVDE	Core High Voltage Detect Enable This field resets only after a POR, LVD, or HVD event. When CORE_HVDE = 1, you must use LP_CFG[BGMODE] to enable the bandgap. If you enable the bandgap to support voltage detection, the low-power mode Idd increases. 0b - Disable 1b - Enable
26 IO_LVDE	IO Low Voltage Detect Enable This field monitors VDD IO on PORT. For port details, see the chip-specific SPC information. This field resets only after a POR, LVD, or HVD event. When IO_LVDE = 1, you must use LP_CFG[BGMODE] to enable the bandgap. If you enable the bandgap to support voltage detection, the low-power mode Idd increases. 0b - Disable 1b - Enable
25 SYS_LVDE	System Low Voltage Detect Enable This field resets only after a POR, LVD, or HVD event. When SYS_LVDE = 1, you must use LP_CFG[BGMODE] to enable the bandgap. If you enable the bandgap to support voltage detection, the low-power mode Idd increases. 0b - Disable 1b - Enable
24 CORE_LVDE	Core Low Voltage Detect Enable This field resets only after a POR, LVD, or HVD event. When CORE_LVDE = 1, you must use LP_CFG[BGMODE] to enable the bandgap. If you enable the bandgap to support voltage detection, the low-power mode Idd increases. 0b - Disable 1b - Enable
23 LP_IREFEN	Low-Power IREF Enable You can disable the low-power IREF only in DPDOWN mode. In all other low-power modes, writing 1 to this field has no effect and the low-power IREF is always enabled. See the chip-specific SPC information for the modules that require the low-power IREF to be enabled in DPDOWN mode when you use the module in DPDOWN mode. 0b - Disable for power saving in Deep Power Down mode 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 —	Reserved
21-20 BGMODE	Bandgap Mode Specifies the bandgap mode configuration. BGMODE will be set to 01b to keep the Bandgap enabled if a write to disable Bandgap is detected while keeping any of the regulators in normal drive strength or if any of the LVD/HVDs are kept enabled or if VDD CORE glitch detect are kept enabled. 00b - Bandgap disabled 01b - Bandgap enabled, buffer disabled 10b - Bandgap enabled, buffer enabled 11b - Reserved
19 —	Reserved
18 LPBUFF_EN	CMP Bandgap Buffer Enable Enables the buffer-stored reference voltage to CMP. The CMP bandgap buffer is automatically disabled and turned off in DPDOWN mode. 0b - Disable 1b - Enable
17 COREVDD_IVS_EN	CORE VDD Internal Voltage Scaling (IVS) Enable Enables the IVS regulator. When enabled, the IVS regulator scales the external input CORE VDD to a lower voltage level to reduce internal leakage during Power Down mode. COREVDD_IVS_EN is automatically 0 in SLEEP or DPDOWN modes. CORE VDD can only get IVS in PDOWN mode. 0b - Disable 1b - Enable
16 —	Reserved
15-13 —	Reserved
12 GLITCH_DETECT_DISABLE	Glitch Detect Disable This field resets only after a POR, LVD, or HVD event.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If you have disabled the bandgap, SPC ignores the value of GLITCH_DETECT_DISABLE and deactivates the glitch-detect hardware. 0b - Enable 1b - Disable
11-10 DCDC_VDD_LVL	DCDC VDD Regulator Voltage Level Specifies the DCDC VDD regulator level. 00b - Retention voltage (0.7 V) 01b - Mid voltage (1.0 V) 10b - Normal voltage (1.1 V) 11b - Overdrive voltage (1.2 V)
9-8 DCDC_VDD_DS	DCDC VDD Drive Strength Specifies the drive strength of the DCDC VDD regulator. If you specify normal drive strength, you must write a value to LP_CFG[BGMODE] that enables the bandgap. The DCDC regulator is always turned off and disabled in DPDOWN mode. Pulse Refresh mode is invalid in SLEEP mode. SPC ignores writes to reserved values. 00b - Pulse refresh 01b - Low 10b - Normal 11b - Reserved
7 —	Reserved
6-5 —	Reserved
4 SYSLDO_VDD_DS	LDO_SYS VDD Drive Strength Specifies the drive strength of the LDO_SYS VDD regulator. If you specify normal drive strength, you must write a value to LP_CFG[BGMODE] that enables the bandgap. If you have enabled LVDs or HVDs, and attempt to specify low drive strength, SPC ignores the attempt. 0b - Low 1b - Normal
3-2	LDO_CORE VDD Regulator Voltage Level

Table continues on the next page...

Table continued from the previous page...

Field	Function
CORELDO_VDD_LVL	Specifies the LDO_CORE VDD regulator level for low-power sleep modes. **NOTE** LDO_CORE overdrive voltage level (1.2V) is not supported when LDO_CORE is configured for low drive. If the CORELDO_VDD_DS fields are set to the same value in both the ACTIVE_CFG and LP_CFG registers, the CORELDO_VDD_LVL's in the ACTIVE_CFG and LP_CFG register must be set to the same voltage level settings. You can change the core VDD levels for the LDO_CORE low power regulator only when ACTIVE_CFG[CORELDO_VDD_DS] = 1. So, before entering any of the low-power states (DSLEEP, PDOWN, DPDOWN) with LDO_CORE low power regulator selected (LP_CFG[CORELDO_VDD_DS] = 0), you must use CORELDO_VDD_LVL to select the correct regulation level during ACTIVE run mode. Updating CORELDO_VDD_LVL sets the SC[BUSY] flag. That flag remains set for at least the total time delay that Active Voltage Trim Delay (ACTIVE_VDELAY) specifies. Before changing CORELDO_VDD_LVL, you must wait until the SC[BUSY] flag clears before entering the selected low-power sleep mode. This ensures the correct core VDD voltage levels are set after the chip is in the chosen mode. Program CORELDO_VDD_LVL to have the same level as LP_CFG[DCDC_VDD_LVL], even if LDO_CORE is off. 00b - Retention voltage 01b - Mid voltage (1.0 V) 10b - Normal voltage (1.1 V) 11b - Overdrive voltage (1.2 V)
1 —	Reserved
0 CORELDO_VDD_DS	LDO_CORE VDD Drive Strength Specifies the drive strength of the LDO_CORE VDD regulator. The LDO_CORE regulator is forced to disabled in DPDOWN Sleep mode. If you specify normal drive strength, you must write a value to LP[BGMODE] that enables the bandgap. If you have enabled LVDs or HVDs, and attempt to specify low drive strength, SPC ignores the attempt. 0b - Low 1b - Normal

37.7.11 Low Power Mode Configuration 1 (LP_CFG1)

Offset

Register	Offset
LP_CFG1	10Ch

Function

Enables different analog modules in Deep Sleep, Power Down, or Deep Power Down modes.

See the chip-specific SPC information for a list of analog modules that this register controls.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SOC_CNTRL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SOC_CNTRL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-0 SOC_CNTRL	Low-Power Configuration Chip Control Provides control bits to the chip to enable analog modules in Low Power mode. Bit combinations with 0: The associated analog modules are disabled. Bit combinations with 1: The associated analog modules are enabled.

37.7.12 Low Power Wake-Up Delay (LPWKUP_DELAY)

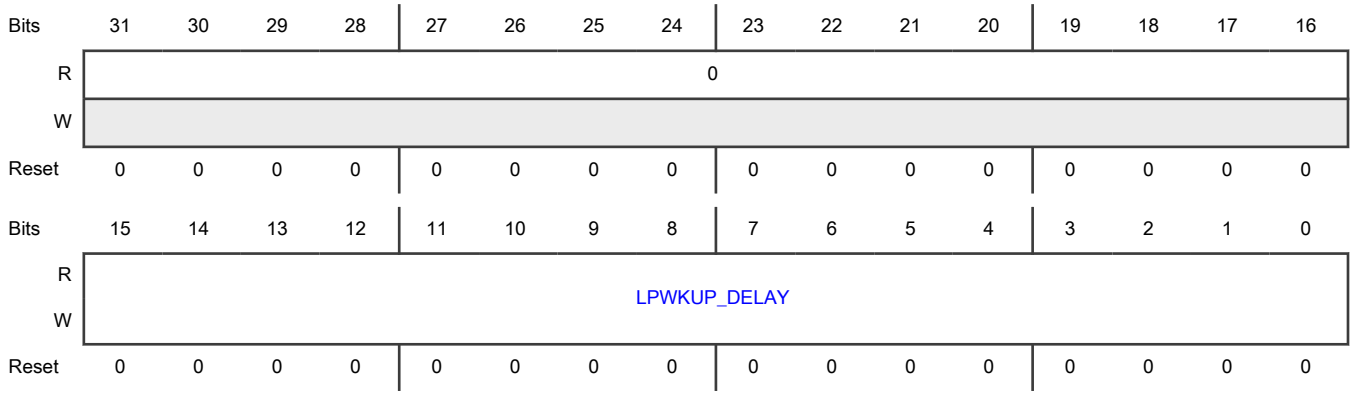
Offset

Register	Offset
LPWKUP_DELAY	120h

Function

Works with the LPREQ pin to extend the wake-up time if the external PMIC or switch needs additional time after wake-up.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 LPWKUP_DELAY	Low-Power Wake-Up Delay Specifies the number of SPC timer clock cycles that SPC waits after exiting low-power modes. This field resets only after a POR, LVD, or HVD event. When voltage levels are not the same between ACTIVE mode and Low Power mode, you must write a nonzero value to this field.

37.7.13 Active Voltage Trim Delay (ACTIVE_VDELAY)

Offset

Register	Offset
ACTIVE_VDELAY	124h

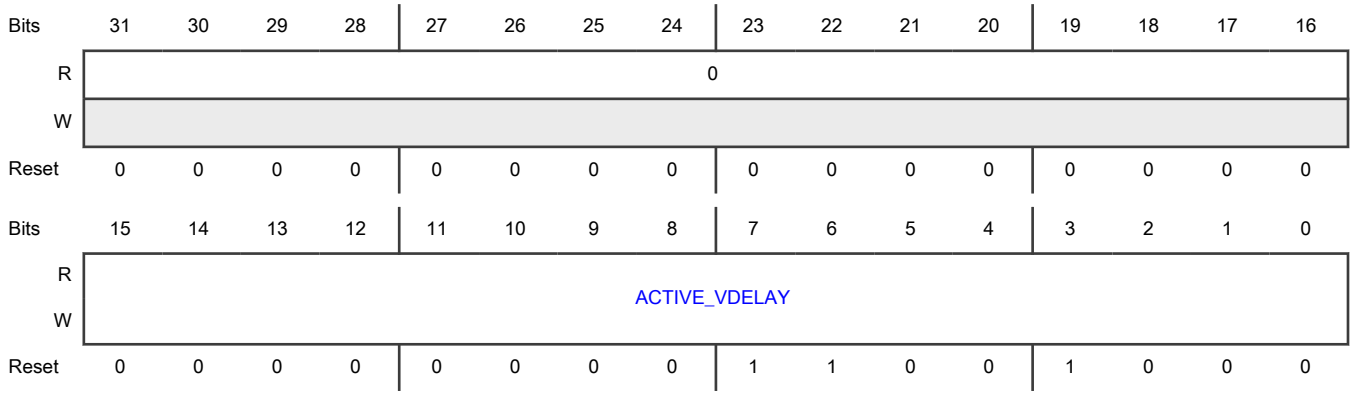
Function

Specifies the delay due to voltage level changes to the regulators in Active mode.

This register resets only after a POR event.

SPC loads the ACTIVE_VDELAY values are loaded from IFR or FUSE after any reset. NXP trims this reset value for the recommended wait time.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 ACTIVE_VDELAY	Active Voltage Delay Specifies the number of SPC timer clock cycles that SPC waits when changing the core-regulator voltage level in Active mode.

37.7.14 Voltage Detect Status (VD_STAT)

Offset

Register	Offset
VD_STAT	130h

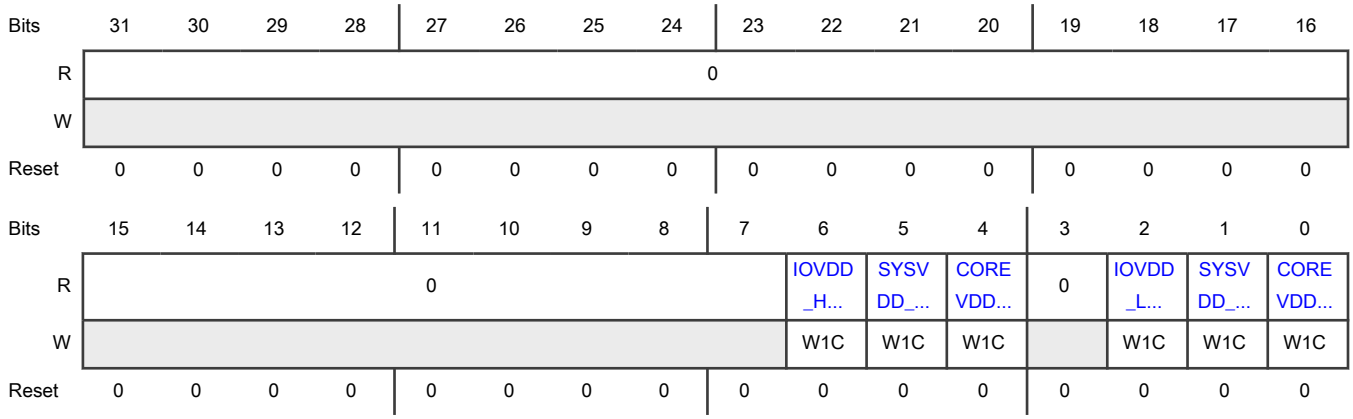
Function

Displays the LVD or HVD captured for:

- Core VDD.
- System VDD.
- IO VDD.

This register resets only after a POR event.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-7 —	Reserved
6 IOVDD_HVDF	IO VDD HVD Flag Indicates an IO VDD HVD event. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Event not detected 1b - Event detected When writing 0b - No effect 1b - Clear the flag
5 SYSVDD_HVDF	System HVD Flag Indicates a system HVD event. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Event not detected 1b - Event detected When writing

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Clear the flag
4 COREVDD_HVDF	Core VDD HVD Flag Indicates a core VDD HVD event. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Event not detected 1b - Event detected When writing 0b - No effect 1b - Clear the flag
3 —	Reserved
2 IOVDD_LVDF	IO VDD LVD Flag Indicates an IO VDD LVD event. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Event not detected 1b - Event detected When writing 0b - No effect 1b - Clear the flag
1 SYSVDD_LVDF	System Low-Voltage Detect Flag Indicates a system LVD event. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Event not detected 1b - Event detected When writing

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Clear the flag
0 COREVDD_LVDF	Core Low-Voltage Detect Flag Indicates a core LVD event. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Event not detected 1b - Event detected When writing 0b - No effect 1b - Clear the flag

37.7.15 Core Voltage Detect Configuration (VD_CORE_CFG)

Offset

Register	Offset
VD_CORE_CFG	134h

Function

Provides these functions:

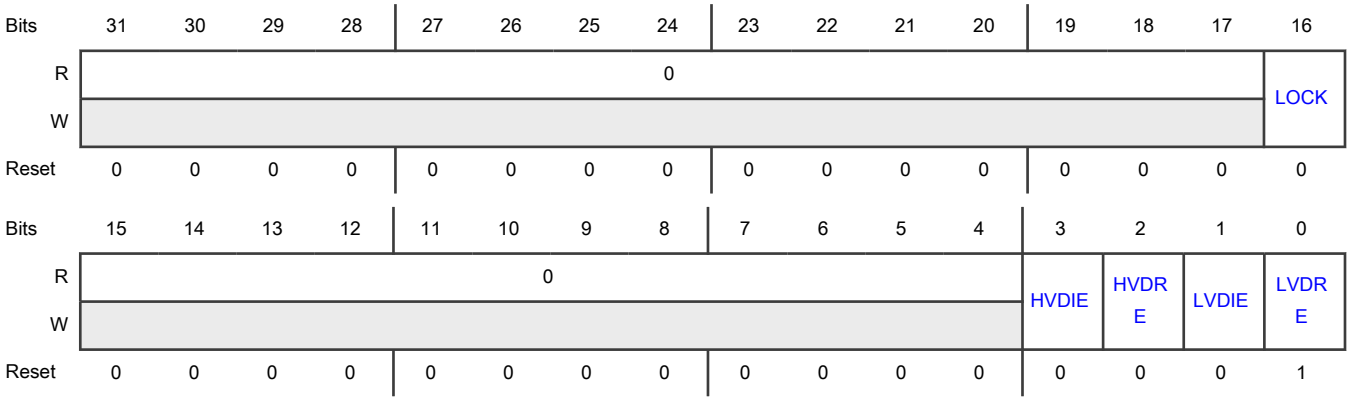
- Configures the low-voltage trip point.
- Enables the core LVD reset and interrupts.

This register resets only after a POR event.

NOTE

If you write 1 to both LVDRE and LVDIE, SPC generates an interrupt after exiting from LVDRE reset. If you don't want this to occur, configure the LVD or HVD so only one is enabled.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 LOCK	Core Voltage Detect Reset Enable Lock Allows writing to the LVDRE and HVDRE fields. 0b - Allow 1b - Deny
15-4 —	Reserved
3 HVDIE	Core VDD HVD Interrupt Enable Enables the Core VDD HVD (COREVDD_HVDF) event to generate a hardware interrupt. If you write 1 to HVDIE with HVD set, SPC automatically generates an HVD interrupt. 0b - Disable 1b - Enable
2 HVDRE	Core VDD HVD Reset Enable Enables the core VDD HVD (COREVDD_HVDF) event to generate a hardware reset. Before writing to this field, you must write 0 to LOCK. 0b - Disable 1b - Enable
1 LVDIE	Core LVD Interrupt Enable Enables the Core LVD (COREVDD_LVDF) event to generate a hardware interrupt. If you write 1 to LVDIE with LVDF set, SPC automatically generates an LVD interrupt.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
0 LVDRE	Core LVD Reset Enable Enables the core LVD (COREVDD_LVDF) event to generate a hardware reset. Before writing to this field, you must write 0 to LOCK. 0b - Disable 1b - Enable

37.7.16 System Voltage Detect Configuration (VD_SYS_CFG)

Offset

Register	Offset
VD_SYS_CFG	138h

Function

Provides these functions:

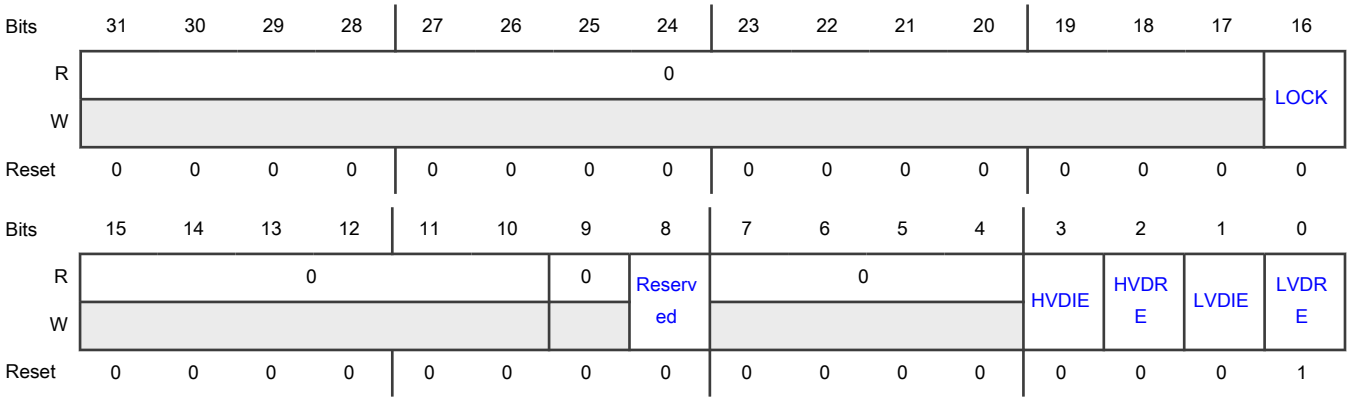
- Configures the low-voltage trip point.
- Enables the system LVD reset and interrupts.

This register resets only after a POR event.

NOTE

If you write 1 to both LVDRE and LVDIE, SPC generates an interrupt after exiting from LVDRE reset. If you don't want this to occur, configure the LVD or HVD so only one is enabled.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 LOCK	System Voltage Detect Reset Enable Lock Allows writing to the LVDRE, HVDRE, and LVSEL fields. 0b - Allow 1b - Deny
15-10 —	Reserved
9 —	Reserved
8 —	Reserved
7-4 —	Reserved
3 HVDIE	System HVD Interrupt Enable Enables the system HVD (SYSVDD_HVDF) event to generate a hardware interrupt. If you write 1 to HVDIE with HVDF set, SPC automatically generates an HVD interrupt. 0b - Disable 1b - Enable
2 HVDRE	System HVD Reset Enable Enables the system HVD (SYSVDD_HVDF) event to generate a hardware reset. Before writing to this field, you must write 0 to LOCK. 0b - Disable 1b - Enable
1 LVDIE	System LVD Interrupt Enable Enables the system LVD (SYSVDD_LVDF) event to generate a hardware interrupt. If you write 1 to LVDIE with LVDF set, SPC automatically generates an LVD interrupt. 0b - Disable 1b - Enable
0	System LVD Reset Enable Enables the system LVD (SYSVDD_LVDF) event to generate a hardware reset.

Table continues on the next page...

Table continued from the previous page...

Field	Function
LVDRE	Before writing to this field, you must write 0 to LOCK. 0b - Disable 1b - Enable

37.7.17 IO Voltage Detect Configuration (VD_IO_CFG)

Offset

Register	Offset
VD_IO_CFG	13Ch

Function

Enables the IO VDD LVD and HVD reset and interrupts.

This register resets only after a POR event.

NOTE

If you write 1 to both LVDRE and LVDIE, SPC generates an interrupt after exiting from LVDRE reset. If you don't want this to occur, configure the LVD or HVD so only one is enabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						0	LVSEL	0				HVDIE	HVDRE	LVDIE	LVDRE
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Fields

Field	Function
31-17 —	Reserved
16	IO Voltage Detect Reset Enable Lock

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOCK	Allows writing to the LVDRE, HVDRE, and LVSEL fields. 0b - Allow 1b - Deny
15-10 —	Reserved
9 —	Reserved
8 LVSEL	IO VDD Low-Voltage Level Select Specifies the LVD trip point for the IO VDD monitor. See the chip data sheet for the high-range and low-range values. Before writing to this field, you must write 0 to LOCK. If you want to change LVSEL, you must do this after disabling the LVD reset and interrupt. Otherwise, SPC could generate an LVD due to the LVSEL change. 0b - High range 1b - Low range
7-4 —	Reserved
3 HVDIE	IO VDD HVD Interrupt Enable Enables the IO VDD HVD (IOVDD_HVDF) event to generate a hardware interrupt. If you write 1 to HVDIE with HVDF set, SPC automatically generates an HVD interrupt. 0b - Disable 1b - Enable
2 HVDRE	IO VDD HVD Reset Enable Enables the IO VDD HVD (IOVDD_HVDF) event to generate a hardware reset. Before writing to this field, you must write 0 to LOCK. 0b - Disable 1b - Enable
1 LVDIE	IO VDD LVD Interrupt Enable Enables the IO VDD LVD (IOVDD_LVDF) event to generate a hardware interrupt. If you write 1 to LVDIE with LVDF set, SPC automatically generates an LVD interrupt. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 LVDRE	IO VDD LVD Reset Enable Enables the IO VDD LVD (IOVDD_LVDF) event to generate a hardware reset. Before writing to this field, you must write 0 to LOCK. 0b - Disable 1b - Enable

37.7.18 External Voltage Domain Configuration (EVD_CFG)

Offset

Register	Offset
EVD_CFG	140h

Function

Isolates signals from external voltage domains under certain conditions.

Chip pins supply the external voltage domains. These domains are controlled at the board level. This register allows you to control internal isolations for signals from these domains if either of the following conditions is true:

- They are not powered on the board.
- They are powered off externally in low-power modes (using the SPC_LPREQ pin).

Any logic in the isolated voltage domain also resets, because the voltage domain is assumed to be powered off. This reset does not impact any VDD_CORE internal power domains or registers—only I/O pads, analog components, or both.

This register resets only after a POR event.

Each bit of the EVDISO, EVDLPISO, and EVDSTAT fields corresponds to one of the I/O or analog supplies according to Table 345.

Table 345. Mapping of bits to supplies

Bit position in field	Description
0	VDD
1	VDD_USB
2	VDD_P2
3	VDD_P3
4	VDD_P4/VDD_ANA
5	Reserved

NOTE

Bit 0 VDD should not be set in EVD_CFG[EVDISO] and it should only be set in EVD_CFG[EVDLPISO] when moving to Deep Power Down mode

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										EVDSTAT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0[1]	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		EVDLPISO						0		EVDISO					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The reset value of this field can change based on external voltage conditions.

Fields

Field	Function
31-22 —	Reserved
21-16 EVDSTAT	External Voltage Domain Status Indicates the status of the external voltage domains. See Table 345 for the mapping of each bit of this field. Each bit's value has the following meaning: 0b - Isolated or not powered 1b - Not isolated
15-14 —	Reserved
13-8 EVDLPISO	External Voltage Domain Low-Power Isolation Isolates the external voltage domain in low-power modes. Use this field if you use the SPC_LPREQ pin to power off any voltage domain in low-power modes. See Table 345 for the mapping of each bit of this field. Each bit's value has the following meaning: 0b - Not isolated 1b - Isolated
7-6 —	Reserved
5-0 EVDISO	External Voltage Domain Isolation Isolates the external voltage domain. Use this field if any voltage domain is always powered off on the board, or whenever a voltage domain is powered off under software control.

Table continues on the next page...

Table continued from the previous page...

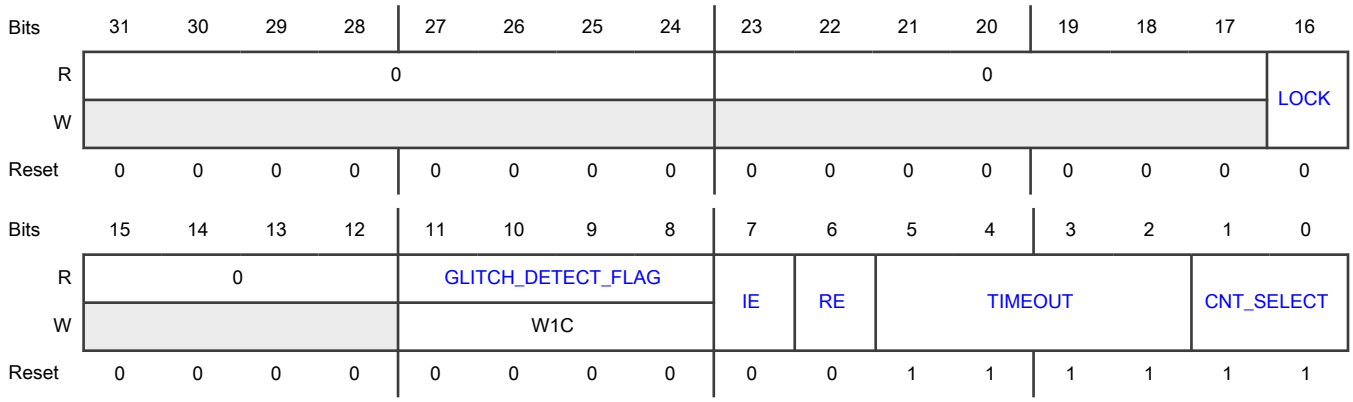
Field	Function
	See Table 345 for the mapping of each bit of this field. Each bit's value has the following meaning: 0b - Not isolated 1b - Isolated

37.7.19 Glitch Detect Status Control (GLITCH_DETECT_SC)

Offset

Register	Offset
GLITCH_DETECT_SC	144h

Diagram



Fields

Field	Function
31-24 —	Reserved
23-17 —	Reserved
16 LOCK	Glitch Detect Reset Enable Lock Bit Used to prevent writes to the GLITCH_RE register bit. 0b - Writes to RE are allowed. 1b - Writes to RE are ignored.
15-12	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
11-8 GLITCH_DETECT_FLAG	GLITCH_DETECT_FLAG State of 4-bit Glitch Ripple Counter output. Based on CNT_SELECT program value, the correct 4-bit output state is monitored to generate interrupt or reset based on the settings of IE and RE.
7 IE	Glitch Detect Interrupt Enable Enables GLITCH_DETECT_FLAG[CNT_SELECT] to generate a hardware interrupt. **NOTE** Setting IE with GLITCH_DETECT_FLAG[CNT_SELECT] set will cause an automatic glitch detect interrupt to get generated. 0b - GLITCH_DETECT_FLAG[CNT_SELECT] does not generate hardware interrupt (user polling) 1b - GLITCH_DETECT_FLAG[CNT_SELECT] does generate hardware interrupt
6 RE	Glitch Detect Reset Enable Enables GLITCH_DETECT_FLAG[CNT_SELECT] to generate a POR/LVD reset. **NOTE** Writes to this bit require the VDD_CORE_GLITCH_DETECT_SC[LOCK] bit to be cleared. 0b - GLITCH_DETECT_FLAG[CNT_SELECT] does not generate POR/LVD reset 1b - GLITCH_DETECT_FLAG[CNT_SELECT] does generate POR/LVD reset
5-2 TIMEOUT	Timeout Specifies the timeout value used to reset glitch detect and compare logic after an initial glitch is detected. The timeout is based on the internal 10 MHz reference clock. After initial glitch is detected, the total timeout used to reset the glitch-detect logic is (TIMEOUT + initialization of internal timer clock).
1-0 CNT_SELECT	Counter Select Specifies the ripple-counter bit to use for detecting a glitch in the VDD core. 00b - 0 01b - 1 10b - 2 11b - 3

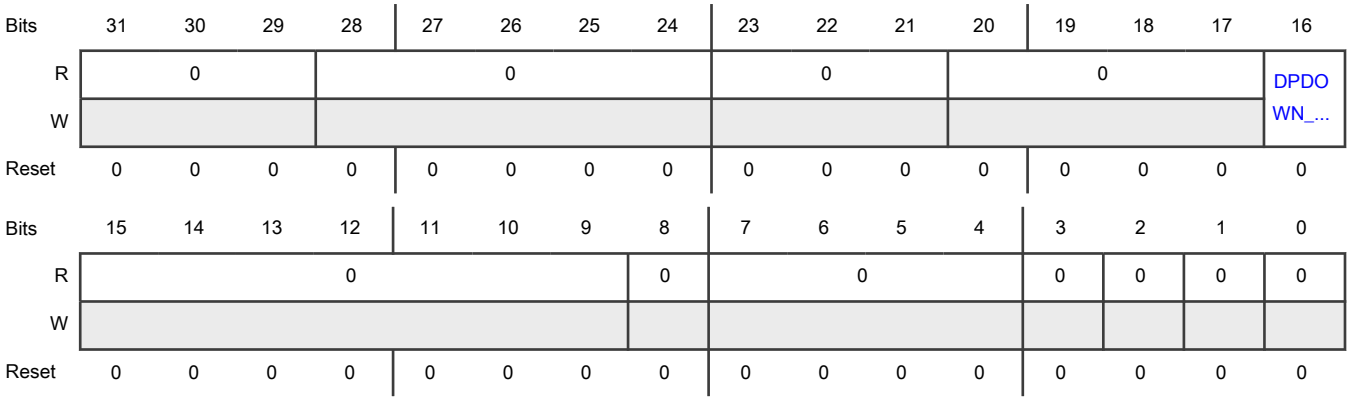
37.7.20 LDO_CORE Configuration (CORELDO_CFG)

Offset

Register	Offset
CORELDO_CFG	300h

Function
Configures LDO_CORE.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-24 —	Reserved
23-21 —	Reserved
20-17 —	Reserved
16 DPDOWN_PULLDOWN_DISABLE	LDO_CORE Deep Power Down Pulldown Disable Forces the LDO_CORE to discharge in Deep Power Down modes if CNTRL[CORELDO_EN] = 1. 0b - LDO_CORE pulldown in Deep Power Down not disabled 1b - LDO_CORE pulldown in Deep Power Down disabled
15-9	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
8 —	Reserved
7-4 —	Reserved
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

37.7.21 LDO_SYS Configuration (SYSLDO_CFG)

Offset

Register	Offset
SYSLDO_CFG	400h

Function

Configures LDO_SYS.

This register resets only after a POR or LVD event.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												0		0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							0	0			0	0			ISINKEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1[1]

1. This value gets loaded with IFR or FUSE values during reset.

Fields

Field	Function
31-20 —	Reserved
19 —	Reserved
18-16 —	Reserved
15-9 —	Reserved
8 —	Reserved
7-5 —	Reserved
4 —	Reserved
3-1 —	Reserved
0 ISINKEN	Current Sink Enable Enables the current-sink feature of the system's low-power regulator. Discharges the output voltage when LDO_SYS is disabled. 0b - Disable 1b - Enable

37.7.22 DCDC Configuration (DCDC_CFG)

Offset

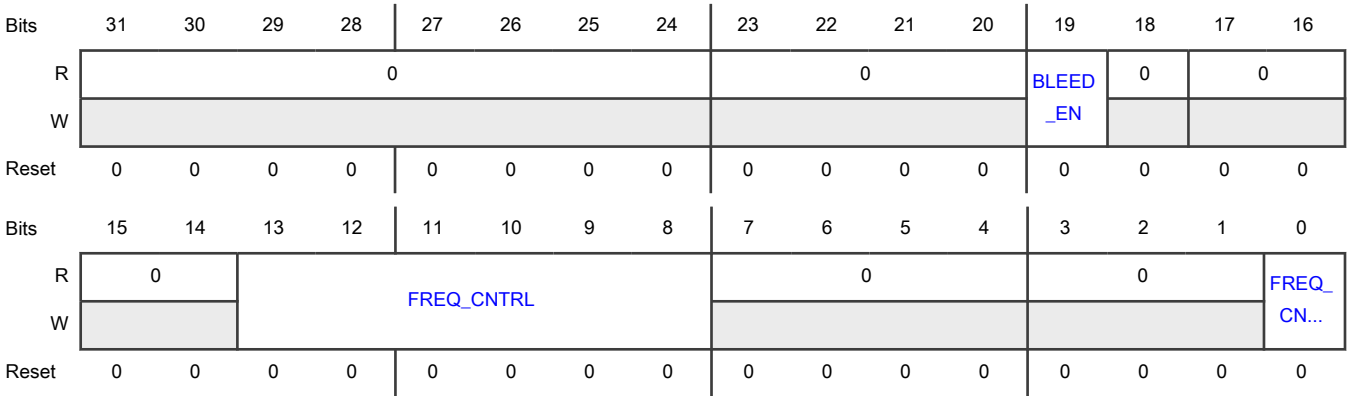
Register	Offset
DCDC_CFG	500h

Function

Configures DCDC.

This register resets only after a POR, LVD, or HVD event.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-20 —	Reserved
19 BLEED_EN	DCDC Bleed Enable Adds a bleed resistor to discharge DCDC output when DCDC is disabled. 0b - Do not add 1b - Add
18 —	Reserved
17-16 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-14 —	Reserved
13-8 FREQ_CNTRL	DCDC Burst Frequency Control Specifies the frequency of the current burst.
7-4 —	Reserved
3-1 —	Reserved
0 FREQ_CNTRL_ON	DCDC Burst Frequency Control Enable Enables DCDC frequency stabilization. 0b - Disable 1b - Enable

37.7.23 DCDC Burst Configuration (DCDC_BURST_CFG)

Offset

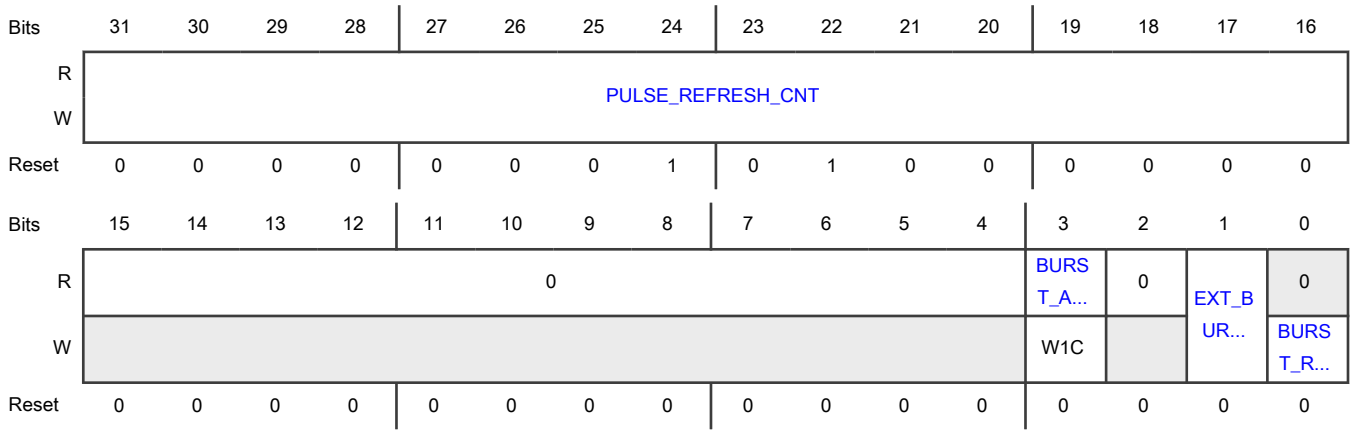
Register	Offset
DCDC_BURST_CFG	504h

Function

Controls the DCDC burst operation.

This register resets only after a POR, LVD, or HVD event.

Diagram



Fields

Field	Function
31-16 PULSE_REFRESH_CNT	Refresh Count Value Controls the DCDC refresh frequency when DCDC is in Pulse Refresh mode. When DCDC is enabled and running in Pulse Refresh mode, PULSE_REFRESH_CNT controls how often the DCDC is pulsed on and off based on a low-power reference clock. You must specify a count value that is long enough to allow a DCDC burst to occur. The pulse duration (time between on and off) is: (reference clock period) × (PULSE_REFRESH_CNT + 2)
15-4 —	Reserved
3 BURST_ACK	Burst Acknowledge Flag Indicates that the DCDC burst completed, and therefore DCDC entered Quiet mode. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not complete 1b - Completed When writing 0b - No effect 1b - Clear the flag
2 —	Reserved
1 EXT_BURST_EN	External Burst Request Enable Enables external burst requests. When this field is 1, SPC asserts the output burst trigger DCDC_BURST_TRIG_PULSE after the DCDC burst request has been acknowledged. **NOTE** Enable external bursts before writing 1 to BURST_REQ and after BURST_ACK is 0. 0b - Disable 1b - Enable
0 BURST_REQ	Software Burst Request Generates a software burst request to DCDC. **NOTE** Do not generate a new burst request until the previous burst request has completed and you acknowledged this by clearing the BURST_ACK flag.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Do not generate 1b - Generate

Chapter 38
Core Mode Controller (CMC)

38.1 Chip-specific CMC information

Table 346. Reference links to related information

Topic	Related module	Reference
Full description	CMC	CMC
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

38.1.1 Module instances

This device has one instance of CMC module, CMC0.

38.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

38.1.3 SRAMDIS0 register

The Reserved fields in CMC_SRAMDIS0 must be written with zero.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W			FLEXSPI0	CAN/ENET/USB1	PQ	USB0	DMA_PKC	LPCAC								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W		RAMH0/1	RAMG2/3	RAMG0/1	RAMF1	RAMF0	RAME/1	RAME0	RAMD1	RAMD0	RAMC1	RAMC0	RAMB	RAMX2	RAMX1	RAMX0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

38.1.4 SRAMRET0 register

The Reserved fields in CMC_SRAMRET0 must be written with zero.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

Table continued from the previous page...

R/W			FLEX SPI0	CAN/ ENET/ USB1	PQ	USB0	DMA_ PKC	LPCAC								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W		RAM H0/1	RAM G2/3	RAM G0/1	RAM F1	RAM F0	RAM E1	RAM E0	RAM D1	RAM D0	RAM C1	RAM C0	RAM B	RAM X2	RAM X1	RAM X0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

38.2 Overview

CMC provides the sequencing of the CPU and associated logic through the different operating modes.

This chapter describes the available CPU operating modes, including reset, active mode, and low-power modes.

38.2.1 Block diagram

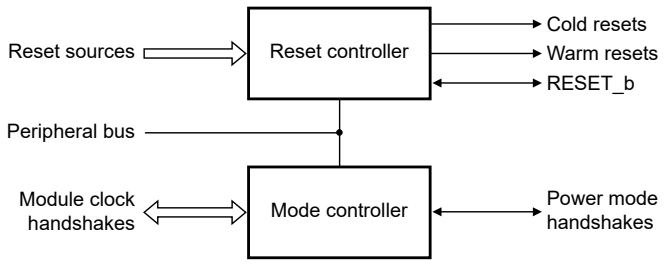


Figure 177. Block diagram

38.2.2 Features

- Configures the low-power mode options including SRAM retention
- Provides reset controller including reset status register and warm reset configuration
- Provides boot mode register and status bits

38.3 Functional description

This section provides the functional description of the block.

38.3.1 Modes of operation

The Arm Cortex-M CPU enters a low-power mode with the execution of a WFI or WFE instruction or via the SLEEPONEXIT mechanism. CMC configures the low-power mode that is entered.

The following table describes the low-power modes available to each core.

Table 347. Operating modes

Mode	Entry	Exit	Description
Reset	POR Warm reset WUU via power down	Active	• Core and peripherals are reset • SRAM optionally retained on power-down wake-up
Active	Reset NVIC WUU	WFI WFE SLEEPONEXIT	Core executing instructions, peripherals optionally active.
Sleep	WFI WFE SLEEPONEXIT	NVIC Reset	• Core waiting for interrupt or event, peripherals optionally active • Core and peripherals retain state. SRAM is active or retained.
Deep Sleep	WFI WFE SLEEPONEXIT	NVIC Reset	• Core waiting for interrupt or event, peripherals optionally active • Core and peripherals retain state, SRAM is retained
Power Down	WFI WFE SLEEPONEXIT	WUU via Active Reset	• Core and peripherals waiting for wakeup • Core and peripherals retain state • SRAM is optionally retained
Deep Power Down	WFI WFE SLEEPONEXIT	WUU via Reset Reset	Core, peripherals, and SRAM not retained.

38.3.2 Active mode

This section describes power options in Active mode.

38.3.2.1 Flash memory

This section describes the Flash Memory Low-Power mode.

You can place the internal flash memory in Low-Power mode under the following conditions:

- Under software control (FLASHCR[FLASHDIS])
- When CPU is sleeping (FLASHCR[FLASHDOZE])
- When the chip enters a low-power mode

When the internal flash memory enters a low-power mode, the wake-up time of the flash memory automatically delays the first access upon exit from Low-Power mode. If there is no access between the internal flash memory exiting Low-Power and subsequent re-entry of Low-Power mode, the wake-up time of the flash memory delays the re-entry.

38.3.2.2 System SRAM

The system SRAM is divided into 32 different power partitions. You can individually power gate these partitions either all the time or only in low-power modes. System SRAM contents in a power gated partition are not retained. You must not access system SRAM partitions when they are power gated.

- Write 1 to SRAMDIS[i] to always power gate system SRAM power partition "i".
- Write 1 to SRAMRET[i] to power gate system SRAM power partition "i" during a low-power mode only.

38.3.3 Low-power modes

This section describes the low-power modes.

38.3.3.1 Low-power entry

The following steps occur during the Low-Power mode entry sequence.

Table 348. Entry sequence

Step	CKMODE (min)	LPMODE (min)	Description
1	0h	0h	Core enters Low-Power mode by WFI/WFE instruction or via SLEEPONEXIT.
2	1h	0h	• Wake-up interrupt controller is enabled. • Core is clock gated.
3	3h	0h	• Request AHB initiators to enter Low-Power mode. • Wait for acknowledge. • AHB initiators are clock gated.
4	3h	0h	• Request AHB peripherals to enter Low-Power mode. • Wait for acknowledge. • AHB peripherals are clock gated.
5	7h	0h	• Request peripherals to enter Low-Power mode. • Wait for acknowledge. • Peripherals are clock gated.
6	Fh	0h	Request power management (regulators, bandgap) to enter Low-Power mode.
7	Fh	1h	• SRAMs enter Low-Power mode (Deep Sleep). • System clocks are gated.
8	Fh	3h	Power domain is placed in Low-Power retention state (Power Down).
9	Fh	7h	Power domain is switched off.
10	Fh	Fh	Regulators are powered off (Deep Power-Down).

38.3.3.2 DMA wakeup

You can configure the DMA to generate a wake-up when a DMA request for configured channel asserts. This is supported when you request the DMA to enter Low-Power mode and retain the DMA state (not in Deep Power-Down).

The DMA wakeup triggers an exit from Low-Power mode for everything except the core, which remains clock gated. Re-enter Low-Power mode after the DMA request negates, and the normal low-power mode entry sequence is followed.

Because a DMA wake-up results in everything except the core from exiting Low-Power mode, you must ensure that other modules not involved in the wakeup remain in a known state. To accomplish this, disable the modules before the initial entry in Low-Power mode or configure the Doze field in selected modules.

If the flash memory is not required during the DMA wakeup, NXP recommends that FLASHCR[FLASHDOZE] = 1.

NOTE

If the DMA request does not negate due to the DMA transfer, the chip remains in a higher power state until an interrupt or other wake-up sources can wakeup the core.

An interrupt that occurs during a DMA wakeup causes an immediate exit from Low-Power mode without affecting the DMA transfer.

38.3.3.3 Power domains

CMC configures Low-Power mode of 2 different power domains when the core enters a low-power mode with CKMODE = Fh. You can configure the different power domains to enter different low-power modes , but you must not configure the WAKE domain to a lower-power mode than any of the other domains.

NOTE

The WAKE domain LPMODE register setting must always be less than or equal to the LPMODE register setting for all other power domains.

CMC controls the following power domains:

- MAIN
- WAKE

The table below lists allowed combinations for MAIN and WAKE.

Table 349. Power Mode Combinations for MAIN and WAKE

CMC PMCTRLMAIN	CMC PMCTRLWAKE
0x0	0x0
0x1	0x1
0x3	0x3
0x3	0x1
0xF	0xF

Configure all power domains to the same Low-Power mode using Global Power Mode Control (GPMCTRL). Alternatively, configure each PMCTRL register to assign an individual low-power mode to each domain.

38.3.3.4 Debug in low-power modes

When the debugger asserts CDBGPWRUPREQ, it requests the debug logic to power up and enable the functionality depending on DBGCTL[SOD].

- When DBGCTL[SOD] = 0, the core clock remains enabled even when the core is sleeping (CKMODE = 0h). CDBGPWRUPACK remains asserted.
- When DBGCTL[SOD] = 1, ignore the debug request when the core sleeps. CDBPWRUPQACK negates when the core is sleeping.

NOTE

Do not attach a debugger when the JTAG/SWD logic is powered down.

38.3.4 Clocks

- The slow bus clock clocks CMC registers and logic.
- clk_1m clock clocks the timeout counters.

38.3.5 Reset

This section describes the sources of reset and the different resets that you can generate.

38.3.5.1 Reset sources

This section describes the different reset sources.

Power-on reset (POR)

When you initially apply power to the MCU or the supply voltage is below the POR falling threshold, the POR circuit triggers the POR condition.

The POR condition asserts a cold reset in all power domains. SRS[POR] and SRS[VD] become 1 on a POR condition.

Voltage detect (VD)

The voltage detect monitors enable by default and keeps the MCU in reset until the supply voltage rises above the LVD rising threshold. Whenever the voltage detect monitors are enabled, they trigger a reset condition if the supply voltage is outside the voltage detect trip point.

SRS[VD] becomes 1 on voltage detect condition. The voltage detect monitors assert a cold reset in all power domains.

Wakeup (WAKEUP)

On a wakeup from Deep-Power Down mode, the power management logic triggers a Wake-up reset in the power domains that had powered off.

The wake-up reset condition asserts a cold reset in all power domains that were powered down. The system power domain, including the RESET_b pin, is unaffected by a wake-up from Deep-Power Down mode (unless another reset source triggers the wake-up). SRS[WAKEUP] becomes 1 on a wake-up reset condition.

External pin reset (RESET_b)

The RESET_b pin is a bidirectional open-drain pin with an internal pull-up resistor. The RESET_b pin function depends on the mode:

- During reset, the RESET_b drives low until the MCU completes initialization, at which point the RESET_b pin is released. If the RESET_b pin asserts externally, then the MCU remains in reset until the RESET_b input is pulled high.
- During active and low-power modes, the RESET_b pin can assert externally to force the MCU into the PIN reset condition.

The RESET_b pin implements a digital filter that you can configure to filter out glitches on the RESET_b pin that are less than 1–32 CMC clock cycles. The filter bypasses in low-power modes when you disable the CMC clock.

The PIN reset condition asserts a warm reset in all power domains. SRS[PIN] and SRS[WARM] become 1 on a PIN reset condition.

Debug access port reset (DAP)

Any debug access port that can initiate a reset request from a connected debugger triggers the DAP reset condition.

The DAP reset condition asserts a warm reset in all power domains. SRS[DAP] and SRS[WARM] become 1 on a DAP reset condition.

Reset timeout (RSTACK)

The reset state machine includes a timeout counter that is triggered by the reset state machine not progressing within 65536 cycles of the clk_1m clock. This triggers the RSTACK reset condition.

The RSTACK reset condition asserts a warm reset in all power domains. SRS[RSTACK], SRS[FATAL], and SRS[WARM] become 1 on an RSTACK reset condition.

Low-power timeout (LPACK)

The low-power entry state machine includes a timeout counter that triggers if a module does not acknowledge entry into Low-Power mode after 65536 cycles of the clk_1m clock. This triggers the LPACK reset condition.

The LPACK reset condition asserts a warm reset in all power domains. SRS[LPACK] and SRS[WARM] become 1 on an LPACK reset condition.

System clock generation (SCG)

The system clock generation includes loss of clock and loss of lock monitors that you can configure to generate a reset, this triggers the SCG reset condition.

The SCG reset condition asserts a warm reset in all power domains. SRS[SCG], SRS[FATAL], and SRS[WARM] become 1 on an SCG reset condition.

Windowed watchdog timer 0 (WWDT0)

The windowed watchdog timer monitors the software by expecting periodic refreshing of the watchdog counter. When this does not occur, it triggers the WWDT0 reset condition.

The WWDT0 reset condition asserts a warm reset in all power domains. SRS[WWDT0] and SRS[WARM] become 1 on a WWDT0 reset condition.

Software (SW)

The software can request a system reset by configuring the system reset request in the Cortex-M33 core, this triggers the software reset condition.

The software reset condition asserts a warm reset in all power domains. SRS[SW] and SRS[WARM] become 1 on a software reset condition.

Lockup (LOCKUP)

The Cortex-M33 core enters Lockup state as a result of certain illegal operations, this triggers the LOCKUP reset condition.

The LOCKUP reset condition asserts an MCU reset in all power domains. SRS[LOCKUP] and SRS[WARM] become 1 on a LOCKUP reset condition.

CPU1 (CPU1)

When CPU1 detects a local reset condition, it triggers the CPU1 reset condition.

The CPU1 reset condition asserts a warm reset in all power domains. SRS[CPU1] and SRS[WARM] becomes 1 on a CPU1 reset condition.

VBAT POR (VBAT)

When VBAT asserts its POR, it triggers the VBAT reset condition.

The VBAT reset condition asserts a warm reset in all power domains. SRS[VBAT] and SRS[WARM] become 1 on a VBAT reset condition.

Windowed watchdog timer 1 (WWDT1)

The windowed watchdog timer monitors the software by expecting periodic refreshing of the watchdog counter. When this does not occur, it triggers the WWDT1 reset condition.

The WWDT1 reset condition asserts a warm reset in all power domains. SRS[WWDT1] and SRS[WARM] become 1 on a WWDT1 reset condition.

Code watchdog (CDOG)

CDOG module helps protect the integrity of software by detecting unexpected changes (faults) in the code execution flow. CDOG module configures to reset or interrupt the processor core when the module detects a fault.

The CDOG0 reset condition asserts a warm reset in all power domains. SRS[CDOG0] and SRS[WARM] become 1 on a CDOG0 reset condition.

The CDOG1 reset condition asserts a warm reset in all power domains. SRS[CDOG1] and SRS[WARM] become 1 on a CDOG1 reset condition.

JTAG reset (JTAG)

When a JTAG instruction places the chip in reset, it triggers the JTAG reset condition.

The JTAG reset condition asserts a warm reset in all power domains. SRS[JTAG]SRS[JTAG], SRS[FATAL]SRS[FATAL], and SRS[WARM] become 1 on a JTAG reset condition.

Security violation (SECVIO)

When a security module detects a security violation, it triggers the SECVIO reset condition.

The SECVIO reset condition asserts a warm reset in all power domains. SRS[SECVIO], SRS[FATAL], and SRS[WARM] become 1 on a SECVIO reset condition.

Tamper detect (TAMPER)

When a tamper detection module detects a tamper event, it triggers the TAMPER reset condition.

The TAMPER reset condition asserts a warm reset in all power domains. The SRS[TAMPER], SRS[FATAL], and SRS[WARM] become 1 on a TAMPER reset condition.

38.3.5.2 Reset types

This section describes the different resets that you can generate.

Cold reset

Each power domain generates a cold reset to reset the debug logic and mode control logic. The cold reset for each power domain can assert as a result of the following events.

- Initial POR of the chip
- Voltage detect monitor (LVD, for example)

- Power domain wake-up from Deep-Power Down (varies per domain)

A cold reset does not guarantee the contents of on-chip SRAM.

Warm reset

- Cold Reset or any of the remaining warm reset sources generates a warm reset.
- A warm reset resets most of the logic in each power domain.
- Warm resets are divided into fatal reset sources and non-fatal reset sources.

You can configure the non-fatal reset sources to generate an interrupt instead of the warm reset. The warm reset averts if you can clear the non-fatal reset source (including status flag) within 65536 cycles of the clk_1m clock. Non-fatal resets retain the contents of on-chip SRAM.

You cannot configure the fatal reset sources to generate an interrupt and do not guarantee the contents of on-chip SRAM.

38.3.5.3 Reset sequence

Power-on reset

Perform the following steps as part of the POR (or voltage detect monitor) sequence:

1. RESET_b pin drives low and internal reset signals asserted.
2. POR or LVD signals negate and clocks are enabled in their default configuration.
3. Internal reset remains asserted for 8 CMC clock cycles.
4. Internal reset to flash memory and fuse controllers negates and their initialization sequences commence.
5. Initialization sequences for flash memory and fuse complete.
6. Internal trim registers are loaded and RESET_b pin is tri-stated.
7. Reset state machine waits for RESET_b pin input to pull high or drive high externally.
8. Internal reset signals negate.
9. Core exits reset and fetches the initial program counter and stack pointer from ROM.

Deep-power down wake-up

Perform the following steps as part of the deep-power down wake-up sequence:

1. Internal reset signals asserted.
2. Wake-up signal negates and clocks are enabled in their default configuration.
3. Internal reset remains asserted for 8 CMC clock cycles.
4. Internal reset to flash memory and fuse controllers negates and commence their initialization sequences.
5. Initialization sequences for flash memory and fuse complete.
6. Internal trim registers are loaded.
7. Internal reset signals negate.
8. Core exits reset and fetches the initial program counter and stack pointer from ROM.

A wake-up from deep-power down via the RESET_b pin follows the warm reset sequence.

Warm reset

Perform the following steps as part of the warm reset sequence:

1. RESET_b pin drives low and internal reset signals asserted.
2. Clocks are enabled in their default configuration.
3. Internal reset remains asserted for 8 CMC clock cycles.
4. Internal reset to flash memory and fuse controllers negates and commence their initialization sequences.
5. Initialization sequences for flash memory and fuse complete.
6. Internal trim registers are loaded and RESET_b pin is tri-stated.
7. Reset state machine waits for RESET_b pin input to pull high or drive high externally.
8. Internal reset signals negate.
9. Core exits reset and fetches the initial program counter and stack pointer from ROM.

38.3.6 Interrupts

CMC generates a single interrupt, which System Reset Interrupt Enable (SRIE) configures.

38.4 External signals

Table 350. External signals

Signal	Description	Direction
RESET_B	Bidirectional open-drain reset pin with pullup that asserts low during warm reset except when waking up from a low-power mode.	Input or output
ISPMODE_n	The input pin is sampled at the end of the RESET_B assertion and stored in Mode (MR0).	Input

38.5 Initialization

By default, the digital glitch filter is disabled on RESET_b pin. To enable the filter, configure:

- RPC[LPFEN], RPC[FILTEN], or both.
- RPC[FILTCFG].

38.6 Application information

To configure for Deep-Power Down Low-Power mode entry:

1. Write Fh to Clock Control (CKCTRL)
2. Write 8h to Power Mode Protection (PMPROT)
3. Write Fh to Global Power Mode Control (GPMCTRL)
4. Execute WFI instruction

To configure for Power-Down Low-Power mode entry:

1. Write Fh to Clock Control (CKCTRL)
2. Write 2h to Power Mode Protection (PMPROT)
3. Write 3h to Global Power Mode Control (GPMCTRL)
4. Execute WFI instruction

To configure for Sleep mode entry:

1. Write 0h or 1h to Clock Control (CKCTRL)
2. Execute WFI instruction

To configure for Deep Sleep Low-Power mode entry:

1. Write Fh to Clock Control (CKCTRL)
2. Write 1h to Power Mode Protection (PMPROT)
3. Write 1h to Global Power Mode Control (GPMCTRL)
4. Execute WFI instruction

To configure for MAIN in power-down and WAKE in Deep Sleep Low-Power mode entry:

1. Write Fh to Clock Control (CKCTRL)
2. Write 3h to Power Mode Protection (PMPROT)
3. Write 3h to PMCTRLMAIN
4. Write 1h to PMCTRLWAKE
5. Execute WFI instruction

38.7 Memory map and register descriptions

This section describes the registers in the CMC module.

38.7.1 CMC register descriptions

NOTE

Different CMC registers reset on different reset types.

NOTE

You must read back the last register written to before executing the WFI instruction. This ensures that before the MCU enters the low power mode, all register writes associated with setting up the low power mode are complete, failure to do this may result in the low power mode not being entered correctly.

38.7.1.1 CMC memory map

CMC0 base address: 4004_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0301_0000h
10h	Clock Control (CKCTRL)	32	RW	0000_0000h
14h	Clock Status (CKSTAT)	32	RW	0000_0000h
18h	Power Mode Protection (PMPROT)	32	RW	0000_0000h
1Ch	Global Power Mode Control (GPMCTRL)	32	RW	0000_0000h
20h	Power Mode Control (PMCTRLMAIN)	32	RW	0000_0000h
24h	Power Mode Control (PMCTRLWAKE)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
80h	System Reset Status (SRS)	32	R	See section
84h	Reset Pin Control (RPC)	32	RW	0000_0000h
88h	Sticky System Reset Status (SSRS)	32	RW	0000_0006h
8Ch	System Reset Interrupt Enable (SRIE)	32	RW	0000_8800h
90h	System Reset Interrupt Flag (SRIF)	32	RW	0000_0000h
9Ch	Reset Count Register (RSTCNT)	32	R	0000_0000h
A0h	Mode (MR0)	32	RW	0000_0000h
B0h	Force Mode (FM0)	32	RW	0000_0000h
C0h	SRAM Disable (SRAMDIS0)	32	RW	0000_0000h
D0h	SRAM Retention (SRAMRET0)	32	RW	0000_0000h
E0h	Flash Control (FLASHCR)	32	RW	0000_0000h
100h	BootROM Status Register (BSR)	32	RW	0000_0000h
10Ch	BootROM Lock Register (BLR)	32	RW	0000_0002h
110h	Core Control (CORECTL)	32	RW	0000_0000h
120h	Debug Control (DBGCTL)	32	RW	0000_0000h

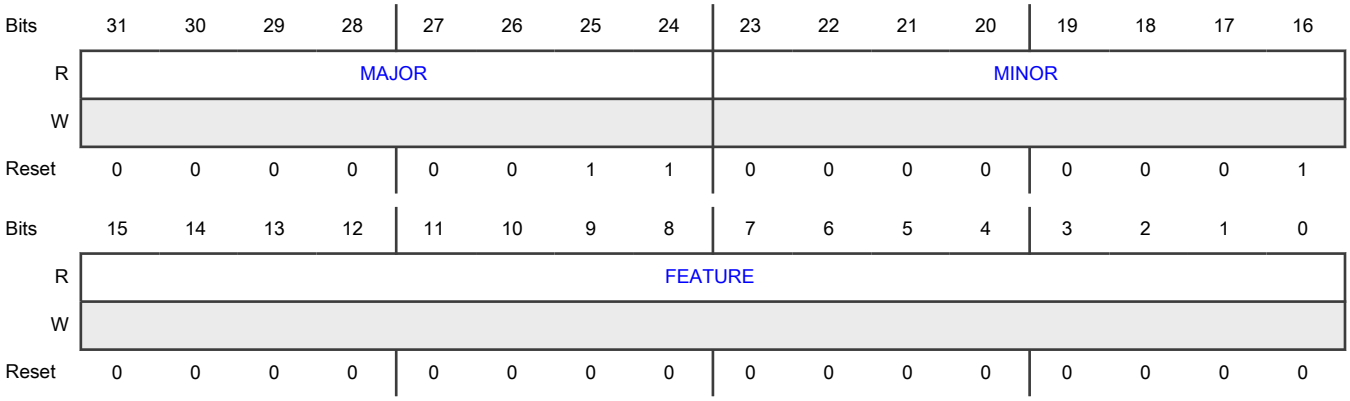
38.7.1.2 Version ID (VERID)

Offset

Register	Offset
VERID	0h

Function
Contains version numbers for module design and feature set.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15-0 FEATURE	Feature Specification Number Returns the feature set number.

38.7.1.3 Clock Control (CKCTRL)

Offset

Register	Offset
CKCTRL	10h

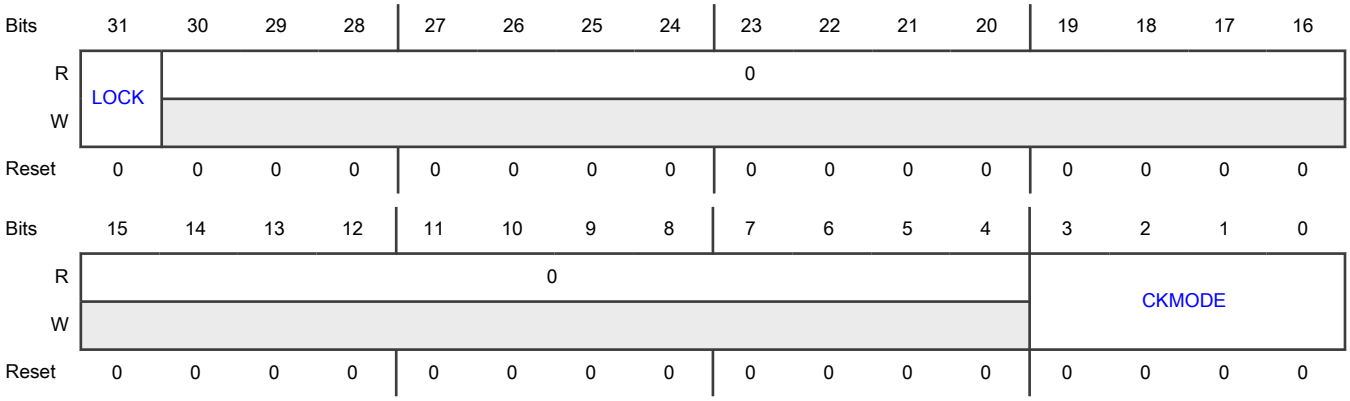
Function

Configures the amount of clock gating when the core asserts sleeping due to WFI, WFE, or SLEEPONEXIT.

NOTE

This register resets on WAKE warm reset.

Diagram



Fields

Field	Function
31 LOCK	Lock Locks the register and blocks writes until the next reset. 0b - Allowed 1b - Blocked
30-4 —	Reserved
3-0 CKMODE	Clocking Mode Configures the amount of clock gating when the core enters a low-power mode because of WFI, WFE or SLEEPONEXIT. Configuring CKMODE > 0 requires the SLEEPDEEP field in the Arm core to become 1. Configuring PMCTRLx[LPMODE] > 0 requires writing Fh to CKMODE. 0000b - Core clock is on 0001b - Core clock is off 1111b - Core, platform, and peripheral clocks are gated, and core enters Low-Power mode.

38.7.1.4 Clock Status (CKSTAT)

Offset

Register	Offset
CKSTAT	14h

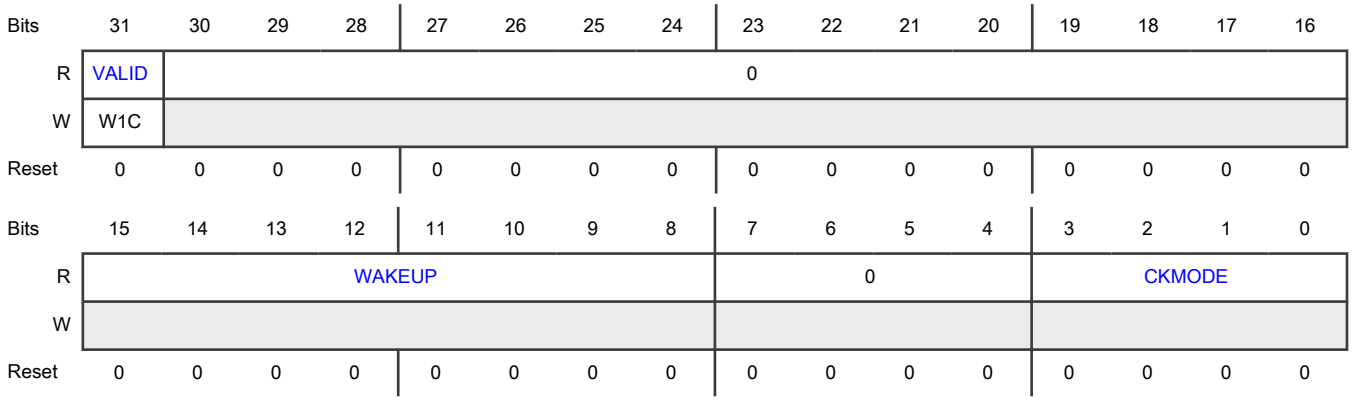
Function

Returns the clock gating status and wake-up source from the previous Low-Power mode entry, provided the core was clock gated. This requires configuring CKCTRL[CKMODE] > 0 and the wake-up event must occur after the core is clock gated. The register contents are only valid when CKSTAT[VALID] = 1.

NOTE

This register resets on WAKE warm reset.

Diagram



Fields

Field	Function
31 VALID	Clock Status Valid Indicates that the core clock was gated since you last wrote 0 to this field. 0b - Core clock not gated 1b - Core clock was gated due to Low-Power mode entry
30-16 —	Reserved
15-8 WAKEUP	Wake-up Source Returns any wake-up sources from the previous Low-Power mode entry. [0] - Wake-up source is reset interrupt or wakeup from Deep Power-Down [1] - Wake-up source is debug request [2] - Wake-up source is interrupt [3] - Wake-up source is DMA wakeup [4] - Wake-up source is WUU request [5] - Wake-up source is USBFS (USB0) [6] - Wake-up source is ITRC [7] - Wake-up source is CPU1 A wakeup from Deep Power-Down writes 1 to WAKEUP[0]. Depending on Low-Power mode, other wake-up bits may not become 1.
7-4 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 CKMODE	Low Power Status Returns the result of the previous Low-Power mode entry. 0000b - Core clock not gated 0001b - Core clock was gated 1111b - Core, platform, and peripheral clocks were gated, and power domain entered Low-Power mode All other values are reserved.

38.7.1.5 Power Mode Protection (PMPROT)

Offset

Register	Offset
PMPROT	18h

Function

Provides protection for entry in low-power modes, you must allow a low-power mode before configuring any Power Mode Control register (PMCTRLx) to that mode.

NOTE

This register resets on WAKE warm reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												LPMODE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Lock Register

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOCK	Locks the register and blocks writes until the next reset. 0b - Allowed 1b - Blocked
30-4 —	Reserved
3-0 LPMODE	Low-Power Mode Bit 0: When set, allows the PMCTRLx[LPM] field to be configured for Deep Sleep. Bit 1: When set, allows the PMCTRLx[LPM] field to be configured for Power Down. Bit 3: When set, allows the PMCTRLx[LPM] field to be configured for Deep Power Down. 0000b - Not allowed 0001b - Allowed 0010b - Allowed 0011b - Allowed 0100b - Allowed 0101b - Allowed 0110b - Allowed 0111b - Allowed 1000b - Allowed 1001b - Allowed 1010b - Allowed 1011b - Allowed 1100b - Allowed 1101b - Allowed 1110b - Allowed 1111b - Allowed

38.7.1.6 Global Power Mode Control (GPMCTRL)

Offset

Register	Offset
GPMCTRL	1Ch

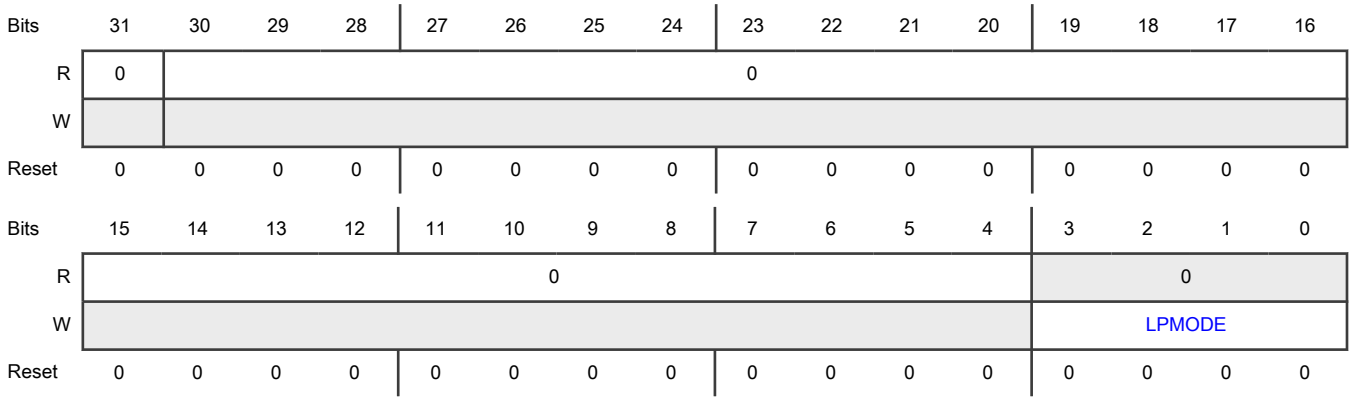
Function

Updates all PMCTRLx registers with the value written.

NOTE

This register resets on WAKE cold reset.

Diagram



Fields

Field	Function
31 —	Reserved
30-4 —	Reserved
3-0 LPMODE	Low-Power Mode Specifies that all PMCTRLx[LPMODE] fields update with the value written.

38.7.1.7 Power Mode Control (PMCTRLMAIN - PMCTRLWAKE)

Offset

Register	Offset
PMCTRLMAIN	20h
PMCTRLWAKE	24h

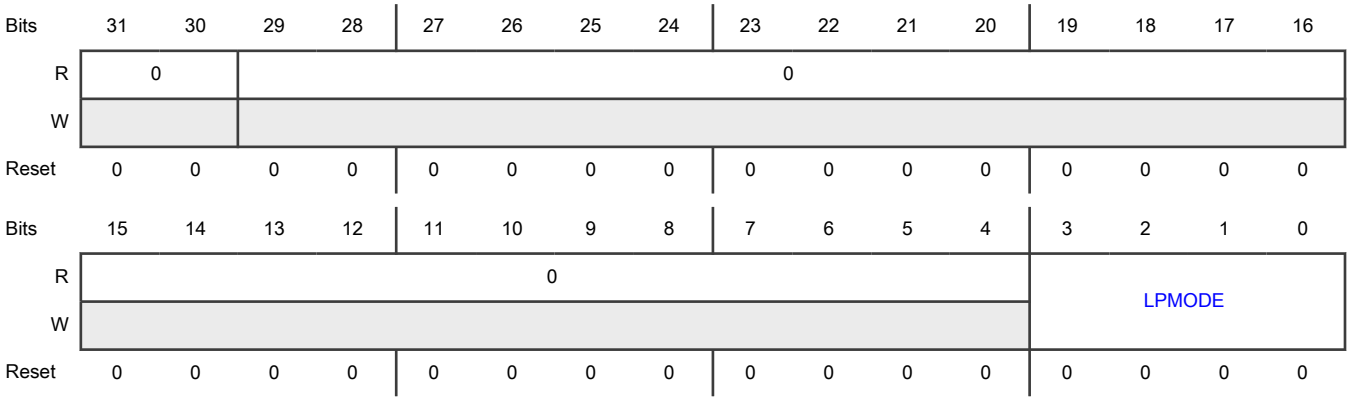
Function

Configures entry into low-power modes for each power domain, provided that the selected power mode is allowed via an appropriate setting of Power Mode Protection (PMPROT), and CKMODE = Fh.

NOTE

This register resets by WAKE cold reset.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-4 —	Reserved
3-0 LPMODE	Low-Power Mode Selects the desired Low-Power mode when a core executes WFI or WFE instruction. Writes to this field are blocked if you have not enabled the protection level using Power Mode Protection (PMPROT). You must not configure this field for the WAKE domain to a lower power mode than any other power domain. **NOTE** See Power domains section for details on allowed combinations. 0000b - Active/Sleep 0001b - Deep Sleep 0011b - Power Down 0111b - Reserved 1111b - Deep-Power Down

38.7.1.8 System Reset Status (SRS)

Offset

Register	Offset
SRS	80h

Function

Updates on every MAIN warm reset to indicate the type/source of the most recent reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAMPER	SECVIO	0	JTAG	CDOG1	CDOG0	WWDT1	VBAT	0	0					0	CPU1
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOCKUP	SW	WWDT0	SCG	LPACK	RSTACK	DAP	PIN	0		FATAL	WARM	0	VD	POR	WAKEUP
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 TAMPER	Tamper Reset Indicates a tamper detection logic that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated
30 SECVIO	Security Violation Reset Indicates a security violation logic that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated
29 —	Reserved
28 JTAG	JTAG System Reset Indicates a JTAG system reset request that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated
27 CDOG1	Code Watchdog 1 Reset Indicates the CDOG1 fault that causes a reset. 0b - Reset is not generated 1b - Reset is generated

Table continues on the next page...

Table continued from the previous page...

Field	Function
26 CDOG0	Code Watchdog 0 Reset Indicates the CDOG0 fault that causes a reset. 0b - Reset is not generated 1b - Reset is generated
25 WWDT1	Windowed Watchdog 1 Reset Indicates the WWDT1 timeout that causes a reset. 0b - Reset is not generated 1b - Reset is generated
24 VBAT	VBAT System Reset Indicates the VBAT POR that generates a system reset. 0b - Reset not generated 1b - Reset generated
23 —	Reserved
22-18 —	Reserved
17 —	Reserved
16 CPU1	CPU1 System Reset Indicates a CPU1 system reset source that generates a reset source. 0b - Reset not generated 1b - Reset generated
15 LOCKUP	Lockup Reset Indicates the Arm core indication of a LOCKUP event that causes a reset. 0b - Reset not generated 1b - Reset generated
14 SW	Software Reset Indicates a software reset request from the Arm core (SYSRESETREQ) that causes a reset. 0b - Reset not generated 1b - Reset generated
13	Windowed Watchdog 0 Reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
WWDT0	Indicates the WWDT 0 timeout that causes a reset. 0b - Reset is not generated 1b - Reset is generated
12 SCG	System Clock Generation Reset Indicates a loss-of-clock or loss-of-lock event in the SCG that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset is not generated 1b - Reset is generated
11 LPACK	Low Power Acknowledge Timeout Reset Indicates a timeout in the Low Power Mode entry logic that causes a reset. This timeout is generated if a peripheral does not acknowledge entry into Low-Power mode within 65536 cycles of clk_1m clock. 0b - Reset not generated 1b - Reset generated
10 RSTACK	Reset Timeout Indicates a timeout or other error condition in the system reset generation logic that causes a reset. This is a fatal reset source, and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated
9 DAP	Debug Access Port Reset Indicates a reset request from a DAP that causes a reset. 0b - Reset was not generated 1b - Reset was generated
8 PIN	Pin Reset Indicates an external assertion of the RESET_b pin that causes a reset. 0b - Reset was not generated 1b - Reset was generated
7-6 —	Reserved
5 FATAL	Fatal Reset Asserts if the last reset source was a fatal reset source. You cannot guarantee SRAM contents following a fatal reset source.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Reset was not generated 1b - Reset was generated
4 WARM	Warm Reset Asserts if the last reset source was a warm reset source. 0b - Reset not generated 1b - Reset generated
3 —	Reserved
2 VD	Voltage Detect Reset Indicates that an enabled voltage detect monitor causes a reset. 0b - Reset not generated 1b - Reset generated
1 POR	Power-on Reset Indicates the POR detection logic that causes a reset. You cannot guarantee SRAM contents following a POR source. 0b - Reset not generated 1b - Reset generated
0 WAKEUP	Wake-up Reset Indicates a wake-up from Deep-Power Down mode that causes a reset. 0b - Reset not generated 1b - Reset generated

38.7.1.9 Reset Pin Control (RPC)

Offset

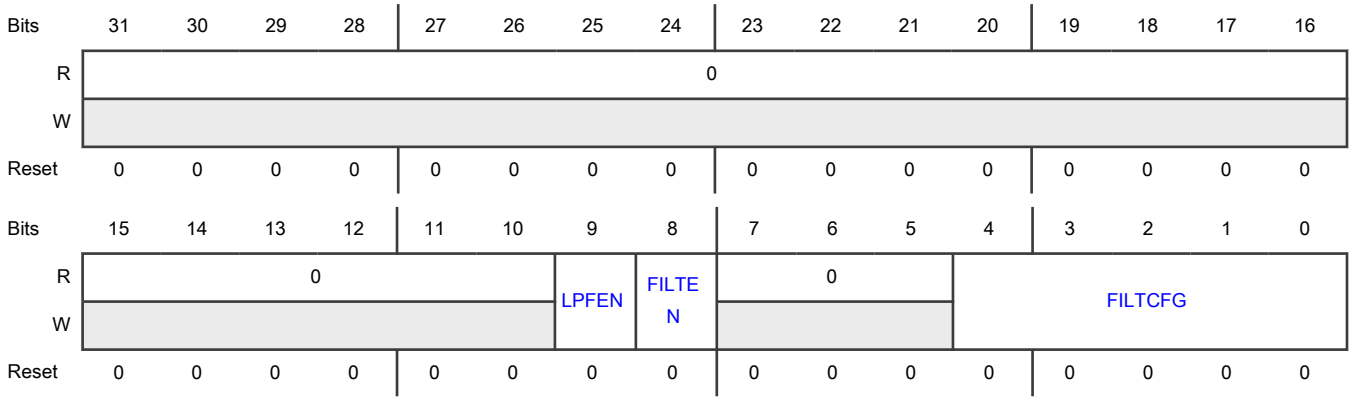
Register	Offset
RPC	84h

Function

NOTE

WAKE cold reset resets this register.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 LPFEN	Low-Power Filter Enable Enables the low-power reset pin filter in both Active and Low-Power modes when this field = 1. The RESET_b pin must assert for more than three clk_1m clock cycles to always propagate through the filter. A glitch less than two clk_1m clock cycles never propagates through the filter. 0b - Disables 1b - Enables
8 FILTEN	Filter Enable Enables the slow clock reset pin filter in Active modes. The RESET_b pin must assert for more than FILTCFG + 1 slow system clock cycles to always propagate through the filter. A glitch less than FILTCFG slow system clock cycles never propagates through the filter. If RPC[LPFEN] also becomes 1, then the filters operate in series. 0b - Disables 1b - Enables
7-5 —	Reserved
4-0 FILTCFG	Reset Filter Configuration Configures the reset pin filter's width from 1 to 32 slow system clock cycles.

38.7.1.10 Sticky System Reset Status (SSRS)

Offset

Register	Offset
SSRS	88h

Function

Stores all sources of system reset that has generated a system reset since the last WAKE cold reset and that you have not cleared. SSRS does not update following a core software reset.

NOTE

This register resets on WAKE cold reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAMPER	SECVIO	0	JTAG	CDOG1	CDOG0	WWDT1	VBAT	0	0					0	CPU1
W	W1C	W1C		W1C	W1C	W1C	W1C	W1C								W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOCKUP	SW	WWDT0	SCG	LPACK	RSTACK	DAP	PIN	0		FATAL	WARM	0	VD	POR	WAKEUP
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C			W1C	W1C			W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Fields

Field	Function
31 TAMPER	Tamper Reset Indicates a tamper detection that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated
30 SECVIO	Security Violation Reset Indicates a security violation that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 —	Reserved
28 JTAG	JTAG System Reset Indicates a JTAG system reset that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated
27 CDOG1	Code Watchdog 1 Reset Indicates a CDOG1 fault that causes a reset. 0b - Reset is not generated 1b - Reset is generated
26 CDOG0	Code Watchdog 0 Reset Indicates a CDOG0 fault that causes a reset. 0b - Reset is not generated 1b - Reset is generated
25 WWDT1	Windowed Watchdog 1 Reset Indicates the WWDT1 timeout that causes a reset. 0b - Reset is not generated 1b - Reset is generated
24 VBAT	VBAT System Reset Indicates the VBAT POR that causes a reset. 0b - Reset not generated 1b - Reset generated
23 —	Reserved
22-18 —	Reserved
17 —	Reserved
16	CPU1 Reset Indicates the CPU1 reset source that causes a reset.

Table continues on the next page...

Table continued from the previous page...

Field	Function
CPU1	0b - Reset not generated from CPU1 reset source. 1b - Reset generated from CPU1 reset source.
15 LOCKUP	Lockup Reset Indicates the core lockup that causes a reset. 0b - Reset not generated 1b - Reset generated
14 SW	Software Reset Indicates the software request from the Arm core (SYSRESETREQ) that causes a reset. 0b - Reset not generated 1b - Reset generated
13 WWDT0	Windowed Watchdog 0 Reset Indicates a windowed watchDog timeout that causes a reset. 0b - Reset is not generated 1b - Reset is generated
12 SCG	System Clock Generation Reset Indicates an SCG loss of lock or loss of clock that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset is not generated 1b - Reset is generated
11 LPACK	Low Power Acknowledge Timeout Reset Indicates a low power acknowledge timeout that causes a reset. 0b - Reset not generated 1b - Reset generated
10 RSTACK	Reset Timeout Indicates a reset controller timeout that causes a reset. This is a fatal reset source and SRAM contents are not guaranteed. 0b - Reset not generated 1b - Reset generated
9 DAP	DAP Reset Indicates a DAP reset request that causes a reset. 0b - Reset not generated

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Reset generated
8 PIN	Pin Reset Indicates a RESET_B pin that causes a reset. 0b - Reset not generated 1b - Reset generated
7-6 —	Reserved
5 FATAL	Fatal Reset Indicates a fatal reset source that causes a reset. 0b - Reset was not generated 1b - Reset was generated
4 WARM	Warm Reset Indicates a warm reset source that causes a reset. 0b - Reset not generated 1b - Reset generated
3 —	Reserved
2 VD	Voltage Detect Reset Indicates that an enabled voltage detect monitor causes a reset. 0b - Reset not generated 1b - Reset generated
1 POR	Power-on Reset Indicates the POR that causes a reset. 0b - Reset not generated 1b - Reset generated
0 WAKEUP	Wake-up Reset Indicates the wake-up from Deep-Power Down mode that causes a reset. 0b - Reset not generated 1b - Reset generated

38.7.1.11 System Reset Interrupt Enable (SRIE)

Offset

Register	Offset
SRIE	8Ch

Function

Delays the assertion of a system reset for 65538 cycles of the clk_1m clock when an interrupt is generated. This allows you to perform a graceful shutdown or to abort the warm reset provided you can clear the pending reset source by resetting the source and then clearing the pending flag. This feature cannot delay a cold or fatal warm reset source. System Reset Status (SRS) updates after the warm reset occurs.

The reset interrupt is not supported in Deep-Power Down mode.

NOTE

This register is reset on WAKE Cold Reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				CDOG1	CDOG0	WWDT1	VBAT	0	0					0	CPU1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOCKUP	SW	WWDT0	SCG	LPACK	0	DAP	PIN	0							
W																
Reset	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27 CDOG1	Code Watchdog 1 Reset 0b - Interrupt disabled 1b - Interrupt enabled
26 CDOG0	Code Watchdog 0 Reset 0b - Interrupt disabled 1b - Interrupt enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
25 WWDT1	Windowed Watchdog 1 Reset 0b - Interrupt disabled 1b - Interrupt enabled
24 VBAT	VBAT System Reset 0b - Interrupt disabled 1b - Interrupt enabled
23 —	Reserved
22-18 —	Reserved
17 —	Reserved
16 CPU1	CPU1 Reset 0b - Interrupt disabled 1b - Interrupt enabled
15 LOCKUP	Lockup Reset **NOTE** A core in the lockup state cannot service interrupts. 0b - Interrupt disabled 1b - Interrupt enabled
14 SW	Software Reset 0b - Interrupt disabled 1b - Interrupt enabled
13 WWDT0	Windowed Watchdog 0 Reset 0b - Interrupt disabled 1b - Interrupt enabled
12 SCG	System Clock Generation Reset 0b - Interrupt disabled 1b - Interrupt enabled
11 LPACK	Low Power Acknowledge Timeout Reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Interrupt disabled 1b - Interrupt enabled
10 —	Reserved
9 DAP	DAP Reset 0b - Interrupt disabled 1b - Interrupt enabled
8 PIN	Pin Reset 0b - Interrupt disabled 1b - Interrupt enabled
7-0 —	Reserved

38.7.1.12 System Reset Interrupt Flag (SRIF)

Offset

Register	Offset
SRIF	90h

Function

Returns the source of the reset interrupt. You can clear the pending reset source by resetting the source and then clearing the pending flag.

NOTE

This register resets on WAKE warm reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				CDOG1	CDOG0	WWDT1	VBAT	0	0					0	CPU1
W					W1C	W1C	W1C	W1C								W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOCKUP	SW	WWDT0	0	LPACK	0	DAP	PIN	0							
W	W1C	W1C	W1C		W1C		W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27 CDOG1	Code Watchdog 1 Reset 0b - Reset source not pending 1b - Reset source pending
26 CDOG0	Code Watchdog 0 Reset 0b - Reset source not pending 1b - Reset source pending
25 WWDT1	Windowed Watchdog 1 Reset 0b - Reset source not pending 1b - Reset source pending
24 VBAT	VBAT System Reset 0b - Reset source not pending 1b - Reset source pending
23 —	Reserved
22-18 —	Reserved
17 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 CPU1	CPU1 Reset 0b - Reset source not pending 1b - Reset source pending
15 LOCKUP	Lockup Reset 0b - Reset source not pending 1b - Reset source pending
14 SW	Software Reset 0b - Reset source not pending 1b - Reset source pending
13 WWDT0	Windowed Watchdog 0 Reset 0b - Reset source not pending 1b - Reset source pending
12 —	Reserved
11 LPACK	Low Power Acknowledge Timeout Reset 0b - Reset source not pending 1b - Reset source pending
10 —	Reserved
9 DAP	DAP Reset 0b - Reset source not pending 1b - Reset source pending
8 PIN	Pin Reset 0b - Reset source not pending 1b - Reset source pending
7-0 —	Reserved

38.7.1.13 Reset Count Register (RSTCNT)

Offset

Register	Offset
RSTCNT	9Ch

Function

The Reset Count Register returns the number of reset sequences completed since the last WAKE Cold Reset.

NOTE

This register is reset on WAKE Cold Reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								COUNT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 COUNT	Count Number of reset sequences completed since the last WAKE Cold Reset. When the count reaches the maximum value, it will saturate and no longer increment.

38.7.1.14 Mode (MR0)

Offset

Register	Offset
MR0	A0h

Function

Contains the state of the boot mode pins sampled at the end of the reset.

NOTE

This register resets on WAKE warm reset, and the reset value may vary depending on the pin state.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															ISPMODE...
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 ISPMODE_n	In System Programming Mode Returns the logic state of the ISPMODE_n pin on the last negation of the RESET_b pin.

38.7.1.15 Force Mode (FM0)

Offset

Register	Offset
FM0	B0h

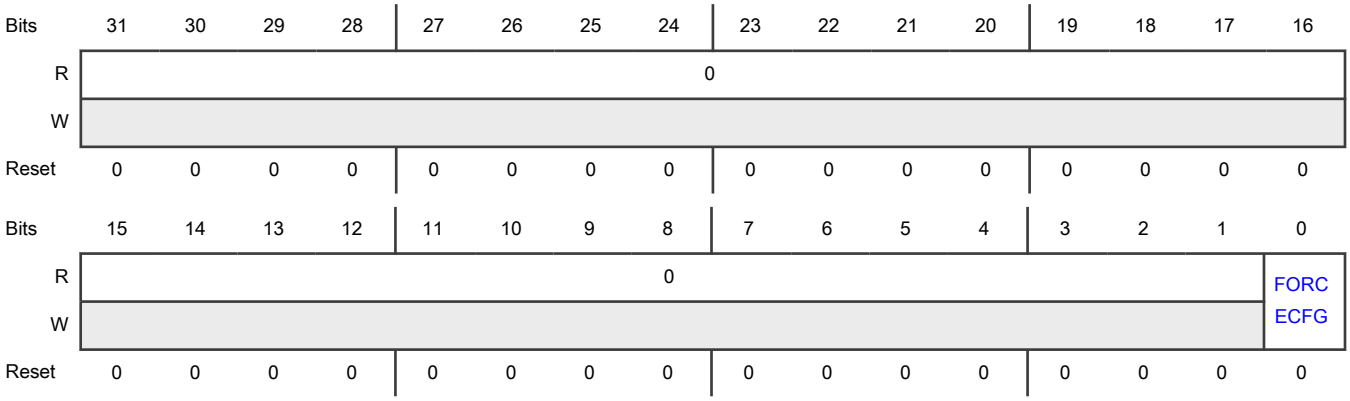
Function

Contains registers that override the state of the boot mode pins.

NOTE

This register resets on WAKE cold reset.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 FORCECFG	Boot Configuration Forces the corresponding field in Mode (MR0) to assert on next system reset. 0b - No effect 1b - Asserts

38.7.1.16 SRAM Disable (SRAMDIS0)

Offset

Register	Offset
SRAMDIS0	C0h

Function

Powers off the specific on-chip System SRAM arrays whenever the corresponding register field becomes 1. You must not access the SRAM array and also do not retain the SRAM array contents.

You must configure this register to 0 before entering DS mode.

NOTE

This register resets on MAIN warm reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DIS31	DIS30	DIS29	DIS28	DIS27	DIS26	DIS25	DIS24	DIS23	DIS22	DIS21	DIS20	DIS19	DIS18	DIS17	DIS16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DIS15	DIS14	DIS13	DIS12	DIS11	DIS10	DIS9	DIS8	DIS7	DIS6	DIS5	DIS4	DIS3	DIS2	DIS1	DIS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DISn	SRAM Disable If this field is 1, corresponding SRAM array is disabled. SRAM array is powered off and must not be accessed. If this field is 0, corresponding SRAM array is enabled. **NOTE** See the chip-specific CMC section for register bit details. 0b - Enables 1b - Disables

38.7.1.17 SRAM Retention (SRAMRET0)

Offset

Register	Offset
SRAMRET0	D0h

Function

Configures if the on-chip System SRAM arrays are retained or powered off in low-power modes.

You must configure this register to 0 before entering DS mode.

NOTE

This register resets on WAKE warm reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RET31	RET30	RET29	RET28	RET27	RET26	RET25	RET24	RET23	RET22	RET21	RET20	RET19	RET18	RET17	RET16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RET15	RET14	RET13	RET12	RET11	RET10	RET9	RET8	RET7	RET6	RET5	RET4	RET3	RET2	RET1	RET0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 RETn	SRAM Retention Controls whether the SRAM array is retained or powered off in Low-Power modes. If this field is 1, corresponding SRAM array is powered off in Low-Power mode. If this field is 0, corresponding SRAM array is retained in Low-Power mode. **NOTE** You cannot retain SRAM arrays in Low-Power modes if their power domain is switched off. See the chip-specific CMC section for register bit details. 0b - Retains 1b - Powers off

38.7.1.18 Flash Control (FLASHCR)

Offset

Register	Offset
FLASHCR	E0h

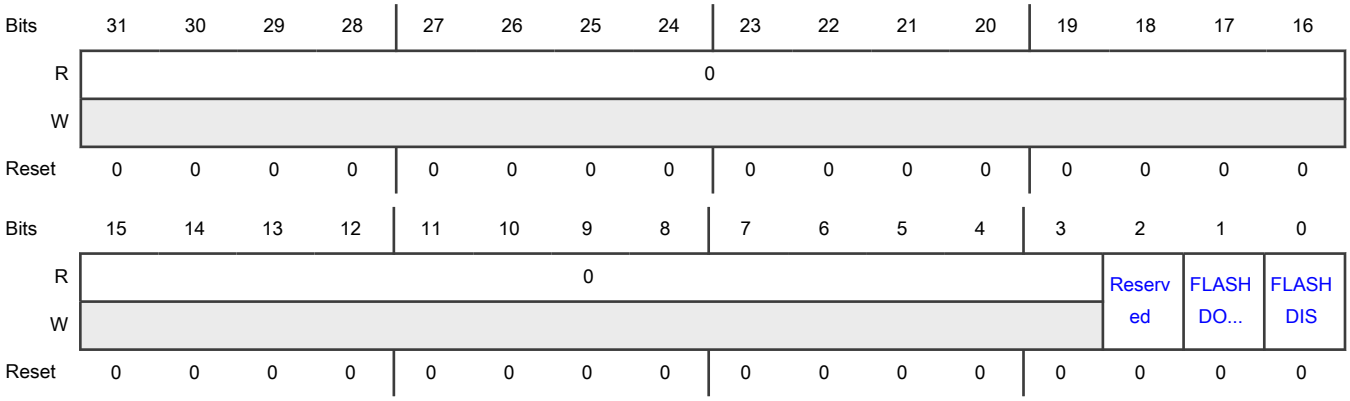
Function

Controls Low-Power mode of the on-chip flash memory.

NOTE

This register resets on WAKE warm reset.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 —	Reserved
1 FLASHDOZE	Flash Doze Disables flash memory accesses and places flash memory in Low-Power state whenever the core clock is gated (CKMODE > 0) because of execution of WFI, WFE, or SLEEPONEXIT. Other bus masters that attempt to access the flash memory stalls until the core is no longer sleeping. 0b - No effect 1b - Flash memory is disabled when core is sleeping (CKMODE > 0)
0 FLASHDIS	Flash Disable Places flash memory in Low-Power state when this field becomes 1. Relocate the interrupts out of flash memory before disabling it. 0b - No effect 1b - Flash memory is disabled

38.7.1.19 BootROM Status Register (BSR)

Offset

Register	Offset
BSR	100h

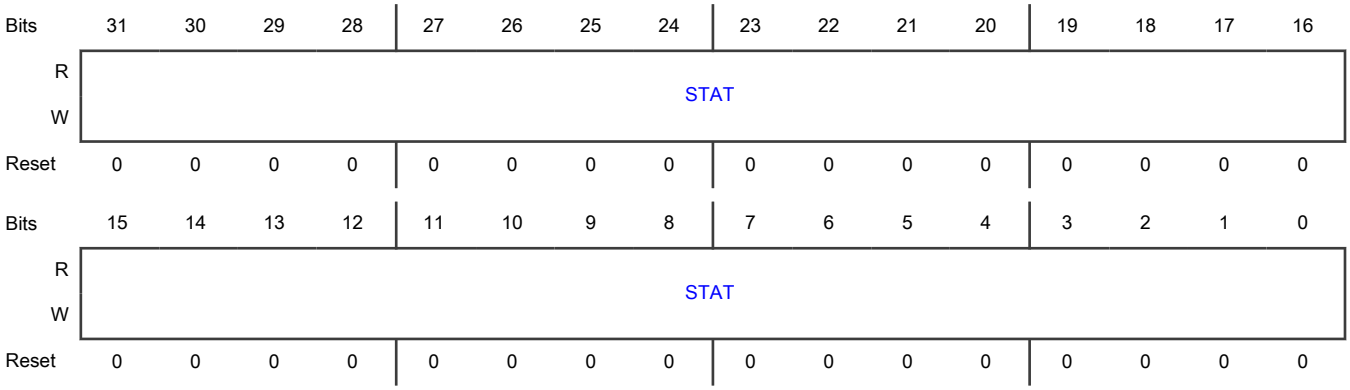
Function

The BootROM Status Register and BootROM Lock Register can only be written when the Lock field equals 010. All other values are reversed and will lock the BootROM registers.

NOTE

This register resets on WAKE cold reset.

Diagram



Fields

Field	Function
31-0 STAT	Provides status information written by the BootROM.

38.7.1.20 BootROM Lock Register (BLR)

Offset

Register	Offset
BLR	10Ch

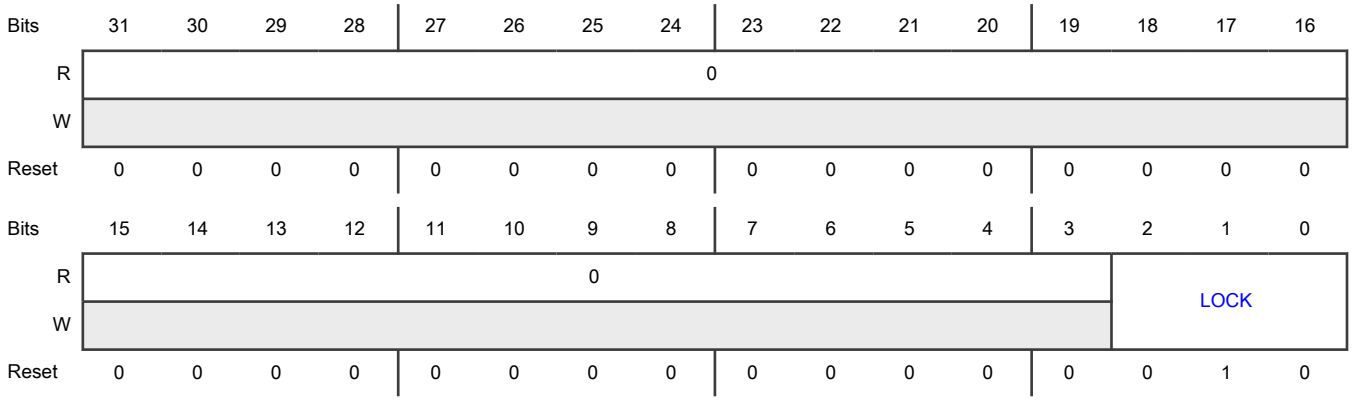
Function

The BootROM Status Register and BootROM Lock Register can only be written when the Lock field equals 010. All other values are reversed and will lock the BootROM registers.

NOTE

This register is reset on MAIN Warm Reset.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 LOCK	Lock The BootROM Status Register and BootROM Lock Register can only be written when the Lock field equals 010. All other values are reversed and will lock the BootROM registers. 010b - BootROM Status and Lock Registers can be written 101b - BootROM Status and Lock Registers cannot be written

38.7.1.21 Core Control (CORECTL)

Offset

Register	Offset
CORECTL	110h

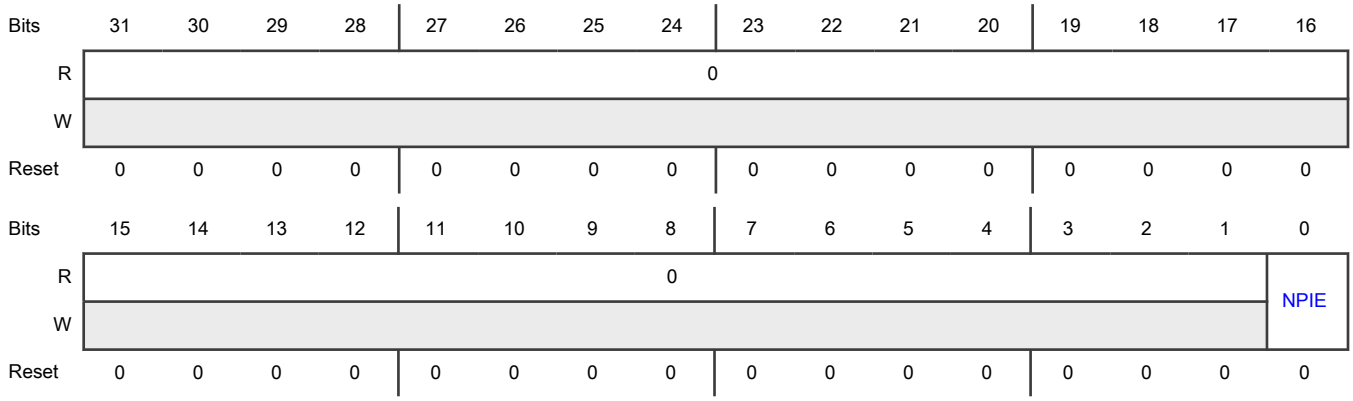
Function

Configures options for the core.

NOTE

This register resets on MAIN warm reset.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 NPIE	Non-maskable Pin Interrupt Enable Enables or disables the pin interrupt. You can write to this field only when NPIE = 0. 0b - Disables 1b - Enables

38.7.1.22 Debug Control (DBGCTL)

Offset

Register	Offset
DBGCTL	120h

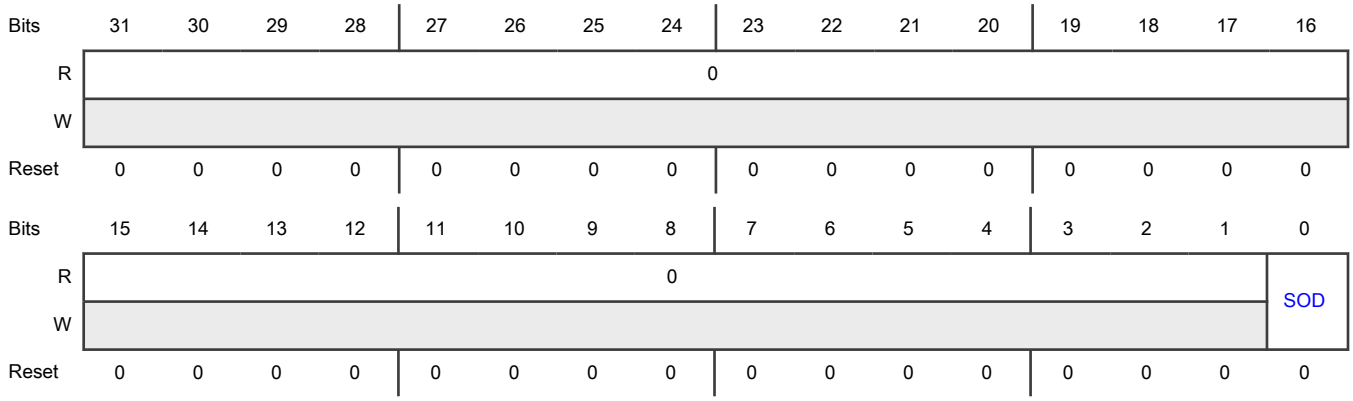
Function

Configures options for debug.

NOTE

This register resets on WAKE cold reset.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 SOD	Sleep Or Debug Configures whether the debug remains enabled when core sleeps. 0b - Remains enabled 1b - Disabled

Chapter 39
Security overview

39.1 Disclaimer

As system security requirements and the attack surface evolves, it is important for customers to understand the types of attacks (especially advanced physical attacks) which NXP does not claim to protect against, or strongly mitigate, so that appropriate mitigation can be taken by the customer at the system level if necessary.

- This SoC has built-in security event detection features. However, NXP does not guarantee against advanced tamper attempts, including operation of the device beyond the defined specification limits. NXP does not guarantee the protection against semi-invasive and invasive attacks.
- This SoC has several built-in features addressing side channel attacks. However, there is no claim to be completely resistant. The effectiveness of these features has not been independently evaluated. Therefore NXP does not guarantee that the result will meet specific customer requirements.
- This SoC's security trust architecture relies on the strength of cryptographic algorithms and digital signatures. If these are subsequently determined to have inherent flaws, then the impact for each flaw must be evaluated and, in this case, NXP does not guarantee the underlying trust architecture claims.
- This SoC has some built-in access control mechanisms to support the logical separation of executed code. However, NXP does not guarantee that the device completely ensures logical separation by itself. Any vulnerabilities identified in Trusted Execution Environments or Hypervisor software may impact this separation and data integrity, and may require additional mitigations.

NXP recommends customers to implement appropriate design and operating safeguards based on defined threat models, to minimize the security risks associated with their applications and products.

39.2 Overview

This chapter provides an overview of Platform Security features the device implements using following on-chip components.

- ELS security subsystem provides key isolation. It also includes random number generation functionality.
- Physically Unclonable Function (PUF) SRAM controller
- PUF that stores the applications keys, which are provisioned to ELS S50 key store, securely.
- Public-key cryptography accelerator (PKC) .
- OTP controller (OTPC) which supports loading and housing of EFUSE content into shadow registers.
- Code Watch Dog (CDOG) to ensure software integrity by detecting unexpected changes in the code execution flow.
- Intrusion and Tamper Response Controller (ITRC) to configure the response action for an intrusion event detected by an on-chip security sensors.
- Memory Block Checker (MBC) that provides read, write, and execute access control per block of internal flash memory.
- Digital tamper (TDET) to support tamper detection.
- Secure AHB bus and AHB Controller to support secure trusted execution at the system level.
- PRINCE in CTR mode of operation to provide confidentiality protection of the content stored on internal flash.
- FlexSPI providing support of inline Prince Encryption Decryption (IPED) in GCM or CTR mode of operation.
- Digital and analog Glitch Detect (GDET) to provide power supply glitch detection.
- Cyclic Redundancy Check (CRC) that provides error detection for multi-bit errors.

The detailed description of each component can be found in the Security Reference Manual for this chip.

NOTE

The EdgeLock Secure Subsystem (ELS) is also known as EdgeLock Secure Enclave, Core Profile (ELE). This document uses the ELS name, but other materials might refer to this module as EdgeLock Secure Enclave, Core Profile or ELE.

The following figure shows the MCX N security system diagram:

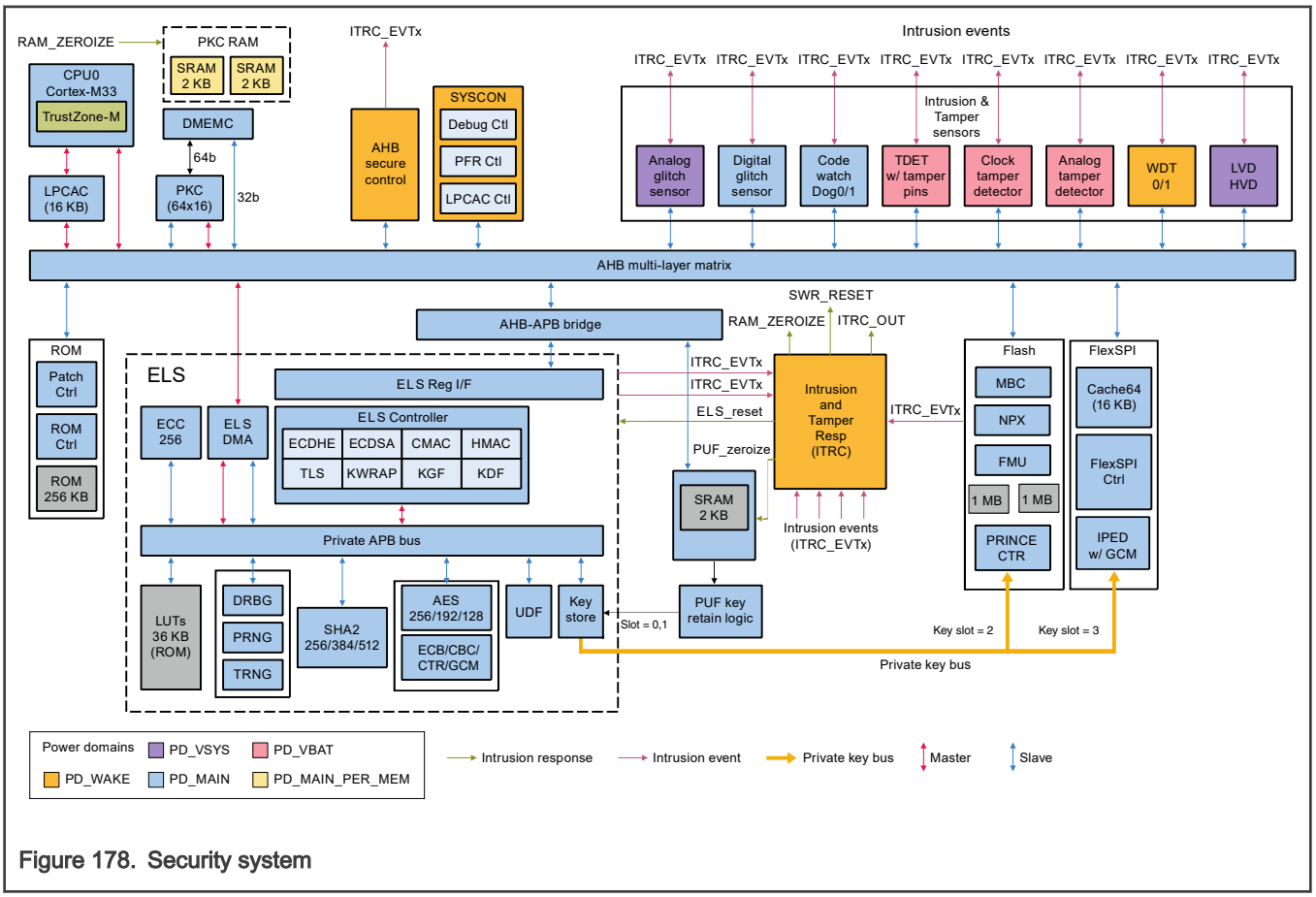


Figure 178. Security system

39.3 Immutable Root of Trust

As defined by Trusted Computing Group, “an Immutable Root of Trust (RoT) is expected to remain identical across all devices within a set of device models based on a defined threat model. It is also expected not to change across time and, therefore, will behave the same during each device’s lifespan.” It consists of truly immutable hardware logic, including analog and digital logic, read-only memory and one-time programmable memory. Immutable RoT is essential for guaranteeing any security feature, including Secure Boot, Secure Debug, Life-cycle Management, and a number of others. In this device, Immutable RoT is embedded in the Boot ROM (immutable bootloader). It uses the device hardware cryptographic functionality for its function.

39.4 Life-cycle management

During its lifespan, a typical device finds itself in various places around the world. It is manufactured in a semiconductor factory, tested and packaged in silicon manufacturer facilities, sold to various distributors, sold further to the Original Equipment Manufacturer (OEM), assembled, tested and provisioned by their Contract Manufacturers and, finally, delivered to the end-customer. In the case of failure, a device is returned to OEM or even back to the silicon manufacturer for further failure analysis.

Device life-cycle state is used to reflect the actual state of the device, which is further used to instruct the device on how exactly to protect the assets a device hosts during specific time. For example, when a device is being tested at a silicon manufacturer facility and no OEM or end-customer assets have been provisioned on it, then access to the device, in terms of debug or test, is less restrictive than when a device is with the end-customer.

Life-cycle state is monotonic, meaning it can only always be increased. Immutable RoT is in charge of life-cycle management and it enforces device access policies accordingly.

39.5 Secure boot

Secure Boot ensures authenticity, integrity and confidentiality of the device bootloader, firmware, and other software during the boot process and ensures that the intended secure life-cycle state is reached. ECDSA P-256 with SHA-256 or ECDSA P-384 with SHA-384 are there to guarantee authenticity and integrity of the firmware image. PRINCE-based memory encryption using a device-unique key derived from PUF can be used to provide code/data confidentiality while the firmware image is stored in internal or external FLASH. Immutable RoT is in charge of enforcing Secure Boot and it does so according to the policies defined by the life-cycle state.

39.6 Secure update

Secure Update is the process used to securely update the firmware image in the field. The firmware image is encrypted using AES-128 or AES-256 and signed using ECDSA P-256 or ECDSA P-384, following the SB3.1 firmware image format. Secure Update guarantees authenticity and confidentiality of the new image. It also ensures that the new image is up-to-date, preventing the rollback to an older image. Running firmware is in charge of receiving and verifying the new firmware image. The follow-up Secure Boot verifies the new firmware image again, making sure the Immutable RoT is still in charge of ensuring authenticity of the latest firmware.

39.7 Secure debug

Secure Debug is the process used to securely access debug, following the policy defined by the life-cycle state. When a debugger wants to debug a device, they indicate that through a so-called debug mailbox and initiate reset. During boot, a device is checking the debug mailbox and starting a full asymmetric-key crypto based challenge-response protocol. A 128-bit random challenge is issued by the device and signed by a debugger. The response is then verified by the device and, if successful, debug is allowed. Immutable RoT is in charge of the whole process.

39.8 IP protection

IP Protection is a set of mechanisms used to protect confidentiality and integrity of valuable code and data stored on the device. During Secure Update, the firmware image is encrypted using AES. While at rest and during Secure Boot, the firmware image can be encrypted using PRINCE. Data or code stored in internal or external memory is encrypted using PRINCE as well. PRINCE encryption is done using a device-unique key, which is derived from PUF by the Immutable RoT. Up to 4 independent memory regions of internal FLASH and up to 4 independent memory regions of external FLASH can be used for protecting IP content from various vendors.

39.9 Secure isolation

Arm TrustZone enables Secure Isolation during run-time by providing four distinct levels of privilege: secure-privilege, secure-user, non-secure-privilege, non-secure-user. Every peripheral is equipped with Peripheral Protection Checker (PPC) that can be programmed to control access to that peripheral, following the Arm TrustZone philosophy. Every memory is equipped with Memory Protection Checker (MPC) that can also be programmed in the same way as the PPC. Secure AHB Controller is in charge of programming all PPC and MPC blocks and only the highest level of privilege, which is secure-privilege, is allowed to do that.

As highlighted by the mechanisms we use for IP Protection, PRINCE-based memory encryption also ensures Secure Isolation between multiple IP vendors. Initial Vector (IV) is derived by secure-privilege and a different value is used for every independent memory region, ensuring the isolation between each other.

39.10 Secure attestation

Secure Attestation is a set of mechanisms used to provide evidence to a remote party on the device’s genuine identity, its software and firmware versions, as well as its integrity and lifecycle state. Device Identity Composition Engine (DICE), as defined by Trusted Computing Group, uses Immutable RoT during boot time to create a unique Device Identity which takes into account Unique Device Secret (UDS), hardware state of the device and its firmware. The DICE feature is implemented using the ELS Runtime

Fingerprint (RTF) in this device. RTF is the NXP-proprietary attestation mechanism, which measures the device's state during boot-time and run-time as well.

39.11 Secure storage

Immutable RoT, including PUF as the source of device-unique master key, is in charge of key derivation and key management. RFC3394 using device-unique key wrapping key is used as a method for securely storing keys. AES GCM using device-unique keys is used for securely storing valuable data. PRINCE memory encryption using device-unique key can be used for storing valuable code in internal and external FLASH.

39.12 Trust provisioning

Trust provisioning is a process used for creation of initial Device Identity keys. Its major objective is to provide a cryptographic proof of the device's origin and to offer a set of tools to OEM for secure provisioning of their own assets. In a nutshell, a device-unique private-public key pair is created on every device, the public portion of which is collected and signed by NXP. That signed public key is installed back onto every device in a form of device-unique certificate, which serves as the actual proof of the device's origin. The corresponding private key, together with other pre-installed key material, is then used for authentication and secure connection to the device, enabling secure provisioning of OEM assets even in a manufacturing environment OEM may not fully trust.

39.13 Secure key management

Secure key management is a process of securing valuable keys and various key material which are essential in maintaining security of the end-user, OEM and NXP assets. The process strongly relies on the Immutable RoT consisting of PUF, hardware logic and ROM. A device-unique master key is provided by PUF and only used for further key derivation. Part of the derivation data is supplied by the Immutable RoT, accurately reflecting the device's state, making sure different keys are derived in different states. For example, different life-cycle state will often yield a different derived key. Similarly, when the debug port is open a different key will be derived than when the debug port is closed. All the platform keys reside within a security subsystem, hidden from the application core at all times.

39.14 Anomaly Detection and Reaction

Anomaly Detection and Reaction describes the processes or algorithms that analyze the device input and output such as sensor data, as well as the software integrity and application operation for abnormal events and, if required, trigger and execute an action. Typically these actions encompass logging the anomaly, issuing a message to the cloud backend, resetting the device, and/or changing a life cycle state.

Chapter 40
Cyclic Redundancy Check (CRC)

40.1 Chip-specific CRC information

Table 351. Reference links to related information

Topic	Related module	Reference
Full description	CRC	CRC
Peripheral memory map		PBRG Memory map
Clocking		Clock distribution
Power management		Power management
Interrupts	NVIC	Interrupts

40.1.1 Module instances

This device has one instance of the CRC module, CRC0.

40.2 Overview

CRC generates 16-bit or 32-bit CRC codes output for error detection. You can calculate these codes for up to 32 bits input data at a time.

CRC provides a programmable polynomial and other parameters to meet 16-bit or 32-bit CRC standards.

40.2.1 Block diagram

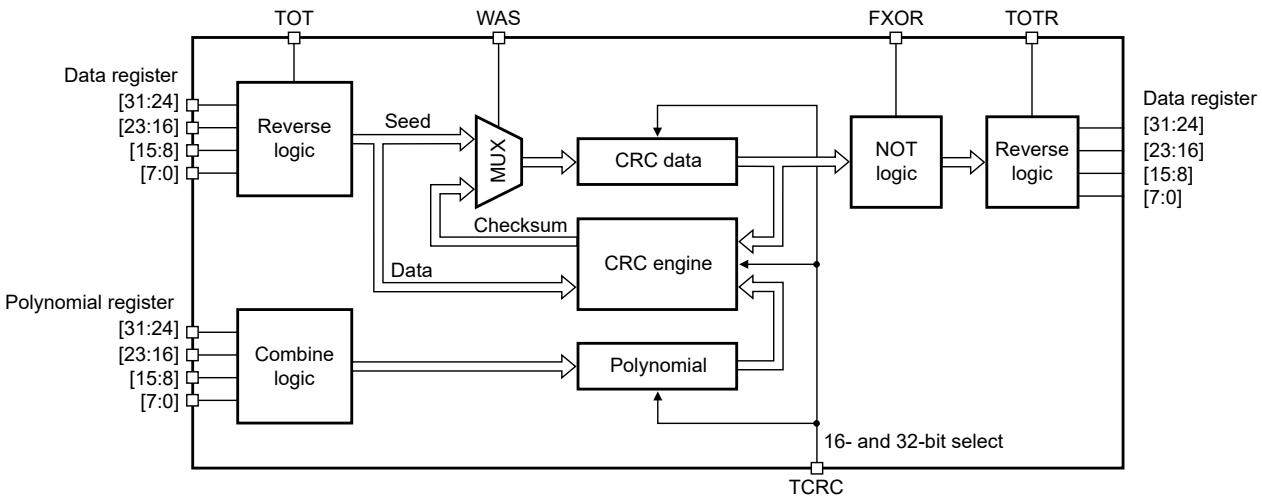


Figure 179. Block diagram

40.2.2 Features

CRC has the following features:

- Hardware CRC generator circuit using 16-bit or 32-bit programmable shift registers

- Programmable initial seed value and polynomial
- Transpose of input or output data (CRC result) in bitwise or bytewise (this option is required for certain CRC standards. You cannot perform a bytewise transpose operation when accessing DATA via 8-bit access.)
- Invert final CRC result
- 32-bit CPU register programming interface

40.3 Functional description

40.3.1 Modes of operation

The following sections describe various modes of operation that affect the functionality of CRC: Run mode and Low power mode.

40.3.1.1 Run mode

Run mode is the basic mode of operation.

40.3.1.2 Low power mode

When the chip enters the lower power mode, the CRC module clock (ipg_clk and ipg_clk_s) is disabled and the in-progress CRC calculation stops. The calculation resumes after the CRC module clock is enabled or the chip exits low power mode via system reset.

40.3.2 CRC calculations

In 16-bit and 32-bit CRC modes, you can program data values as 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous. Noncontiguous bytes lead to an incorrect CRC calculation.

40.3.2.1 Calculating a 16-bit CRC

Perform these steps to calculate a 16-bit CRC:

1. Write 0 to CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in Data (DATA) as required for the CRC calculation.
3. Write a 16-bit polynomial to GPOLY[LOW].

   GPOLY[HIGH] is not usable in 16-bit CRC mode.
4. Write 1 to CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to DATA[LU] and DATA[LL].

   DATA[HU] and DATA[HL] are not used.
6. Write 0 to CTRL[WAS] to start writing data values.
7. Write data values into DATA[LU], and DATA[LL]

   CRC is calculated on every data value write and the intermediate CRC result is stored back into DATA[LU] and DATA[LL]
8. After writing all the data values, read the final CRC result from DATA[LU] and DATA[LL].

CRC is calculated bytewise and two clocks are required to complete one CRC calculation.

The transpose and complement operations are performed on-the-fly when reading or writing values. See Transpose feature and Result complement for details.

40.3.2.2 Calculating a 32-bit CRC

Perform these steps to calculate a 32-bit CRC:

1. Write 1 to CTRL[TCRC] to enable 32-bit CRC mode.

2. Program the transpose and complement options in Control (CTRL) as required for CRC calculation. See Transpose feature and Result complement for details.
3. Write a 32-bit polynomial to GPOLY[HIGH] and GPOLY[LOW] .
4. Write 1 to CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to DATA[HU], DATA[HL], DATA[LU], and DATA[LL].
6. Write 0 to CTRL[WAS] to start writing data values.
7. Write data values into DATA[HU], DATA[HL], DATA[LU], and DATA[LL]

   CRC is computed on every data value write and the intermediate CRC result is stored back into DATA[LU] and DATA[LL]
8. After writing all the values, read the final CRC result from DATA[HU], DATA[HL], DATA[LU], and DATA[LL].

   CRC is calculated bytewise and two clocks are required to complete one CRC calculation.

The transpose and complement operations are performed on-the-fly when reading or writing values. See Transpose feature and Result complement for details.

40.3.3 Transpose feature

Transpose is not enabled by default. However, CRC requires input data and/or final checksum to be transposed. You have an option to configure each transpose operation separately to meet CRC standards. The data is transposed on-the-fly while being read or written.

Some protocols use the little-endian format for data stream to calculateCRC. In this case, transpose flips bits.

40.3.3.1 Types of transpose

CRC provides several types of transpose to flip bits and/or bytes for both writing input data and reading result separately using the CTRL[TOT] and CTRL[TOTR] according to the CRC calculation being used.

The following types of transpose are available for writing to and reading from DATA.

1. CTRL[TOT] or CTRL[TOTR] is 0.

   No transposition occurs.
2. CTRL[TOT] or CTRL[TOTR] is 1.

   Bits in a byte are transposed when bytes are not transposed.

   reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}.

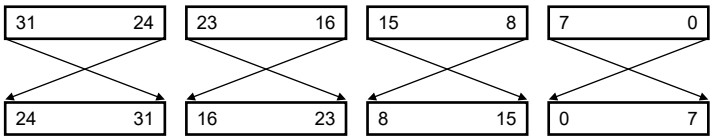


Figure 180. Transpose type 1b

3. CTRL[TOT] or CTRL[TOTR] is 10b.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes {reg[0:7], reg[8:15], reg[16:23], reg[24:31]}.

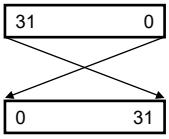


Figure 181. Transpose type 10b

4. CTRL[TOT] or CTRL[TOTR] is 11b.

   Bytes are transposed but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}.

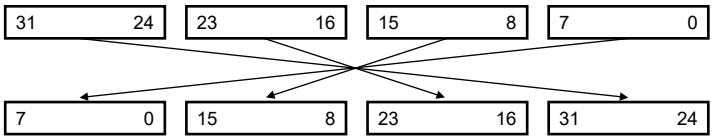


Figure 182. Transpose type 11b

NOTE

For 8-bit and 16-bit write accesses to Data (DATA), the data is transposed with 0s on the unused byte or bytes (taking 32 bits as a whole), but CRC is calculated on the valid byte(s) only. When reading the Data (DATA) for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in DATA[HU] and DATA[HL]. You must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

40.3.4 Result complement

When CTRL[FXOR] = 1, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in Data (DATA) every time Data (DATA) is read. When CTRL[FXOR] = 0,, reading Data (DATA) accesses the raw checksum value.

40.3.5 Clocking

Table 352. CRC clocks

Type of clock	Description
Bus clock (ipg_clk/ipg_clk_s)	ipg_clk_s controls the access to the CRC registers. ipg_clk and ipg_clk_s function the CRC module.

40.3.6 Interrupts

This module has no interrupts.

40.4 External signals

There is no CRC signal that connects off chip.

40.5 Initialization

To enable CRC calculation, you must program:

- CTRL[WAS] .
- Polynomial (GPOLY).

- Parameters for transposition and CRC result inversion in the applicable registers.

Writing 1 to CTRL[WAS] enables you to program the seed value into CRC Data registers.

After writing all the data, you must wait for at least two clock cycles to read the data from CRC Data (DATA) register.

After a CRC calculation completes, you can reinitialize the module for a new CRC computation by again writing 1 to CTRL[WAS] and programming a new, or previously used, seed value. You must set all other parameters before programming the seed value and subsequent data values.

40.6 Use cases

The following tables use the little-endian format.

40.6.1 CTRL programming

The following table shows Control (CTRL) programming for 16-bit CRC.

Table 353. CTRL programming for 16-bit CRC

Algorithm	Polynomial	Seed	Ref in	Ref out	XOR out	CTRL[TOT]	CTRL[TOTR]	CTRL[FXOR]
CRC-16_CCITT_FALSE	1021h	FFFFh	0	0	0000h	0h	0h	0h
CRC-16_ARC	8005h	0000h	1	1	0000h	1h	2h	0h
CRC-16_AUG_CCITT	1021h	1D0Fh	0	0	0000h	0h	0h	0h
CRC-16_BUYPASS	8005h	0000h	0	0	0000h	0h	0h	0h
CRC-16_CCITT_ZERO	1021h	0000h	0	0	0000h	0h	0h	0h
CRC-16_CDMA2000	C867h	FFFFh	0	0	0000h	0h	0h	0h
CRC-16_DDS_110	8005h	800Dh	0	0	0000h	0h	0h	0h
CRC-16_DECT_X	589h	0000h	0	0	0000h	0h	0h	0h
CRC-16_DNP	3D65h	0000h	1	1	FFFFh	1h	2h	1h
CRC-16_EN_13757	3D65h	0000h	0	0	FFFFh	0h	0h	1h
CRC-16_GENIBUS	1021h	FFFFh	0	0	FFFFh	0h	0h	1h
CRC-16_MAXIM	8005h	0000h	1	1	FFFFh	1h	2h	1h
CRC-16_MCRF4XX	1021h	FFFFh	1	1	0000h	1h	2h	0h
CRC-16_RIELLO	1021h	B2AAh	1	1	0000h	1h	2h	0h
CRC-16_T10_DIF	8BB7h	0000h	0	0	0000h	0h	0h	0h
CRC-16_TELEDISK	A097h	0000h	0	0	0000h	0h	0h	0h
CRC-16_TMS37157	1021h	89ECh	1	1	0000h	1h	2h	0h
CRC-16_USB	8005h	FFFFh	1	1	FFFFh	1h	2h	1h
CRC-16_A	1021h	C6C6h	1	1	0000h	1h	2h	0h
CRC-16_KERMIT	1021h	0000h	1	1	0000h	1h	2h	0h
CRC-16_MODBUS	8005h	FFFFh	1	1	0000h	1h	2h	0h
CRC-16_X_25	1021h	FFFFh	1	1	FFFFh	1h	2h	1h
CRC-16_XMODEM	1021h	0000h	0	0	0000h	0h	0h	0h

The following table shows Control (CTRL) programming for 32-bit CRC.

Table 354. CTRL programming for 32-bit CRC

Algorithm	Polynomial	Seed	Ref in	Ref out	XOR out	CTRL[TOT]	CTRL[TOTR]	CTRL[FXOR]
CRC-32	04C11DB7h	FFFFFFFFh	1	1	FFFF_FFFFh	1h	2h	1h
CRC-32_BZIP2	04C11DB7h	FFFFFFFFh	0	0	FFFF_FFFFh	0h	0h	1h
CRC-32C	1EDC6F41h	FFFFFFFFh	1	1	FFFF_FFFFh	1h	2h	1h
CRC-32D	A833982Bh	FFFFFFFFh	1	1	FFFF_FFFFh	1h	2h	1h
CRC-32_MPEG-2	04C11DB7h	FFFFFFFFh	0	0	0000_0000h	0h	0h	0h
CRC-32_POSIX	04C11DB7h	00000000h	0	0	FFFF_FFFFh	0h	0h	1h
CRC-32Q	814141ABh	00000000h	0	0	0000_0000h	0h	0h	0h
CRC-32_JAMCRC	04C11DB7h	FFFFFFFFh	1	1	0000_0000h	1h	2h	0h
CRC-32_XFER	000000AFh	00000000h	0	0	0000_0000h	0h	0h	0h

40.6.2 Expected read data fields

The following table shows the expected read data fields for 16-bit CRC.

Table 355. Expected read data fields for 16-bit CRC

Algorithm	Data (DATA)
CRC16_CCITT_FALSE	[31:16] = Unknown[15:0] = Valid data
CRC16_ARC	[31:16] = Valid data [15:0] = Unknown
CRC16_AUG_CCITT	[31:16] = Unknown [15:0] = Valid data
CRC16_BUYPASS	[31:16] = Unknown [15:0] = Valid data
CRC16_CCITT_ZERO	[31:16] = Unknown [15:0] = Valid data
CRC16_CDMA2000	[31:16] = Unknown [15:0] = Valid data
CRC16_DDS_110	[31:16] = Unknown [15:0] = Valid data
CRC16_DECT_X	[31:16] = Unknown [15:0] = Valid data
CRC16_DNP	[31:16] = Valid data [15:0] = Unknown
CRC-16_EN_13757	[31:16] = Unknown [15:0] = Valid data
CRC-16_GENIBUS	[31:16] = Unknown [15:0] = Valid data
CRC-16_MAXIM	[31:16] = Valid data [15:0] = Unknown
CRC-16_MCRF4XX	[31:16] = Valid data [15:0] = Unknown
CRC-16_RIELLO	[31:16] = Valid data [15:0] = Unknown
CRC-16_T10_DIF	[31:16] = Unknown [15:0] = Valid data
CRC-16_TELEDISK	[31:16] = Unknown [15:0] = Valid data
CRC-16_TMS37157	[31:16] = Valid data [15:0] = Unknown

Table continues on the next page...

Table 355. Expected read data fields for 16-bit CRC (continued)

Algorithm	Data (DATA)
CRC-16_USB	[31:16] = Valid data [15:0] = Unknown
CRC-16_A	[31:16] = Valid data [15:0] = Unknown
CRC-16_KERMIT	[31:16] = Valid data [15:0] = Unknown
CRC-16_MODBUS	[31:16] = Valid data [15:0] = Unknown
CRC-16_X_25	[31:16] = Valid data [15:0] = Unknown
CRC-16_XMODEM	[31:16] = Unknown [15:0] = Valid data

The following table shows the expected read data fields for 32-bit CRC.

Table 356. Expected read data fields for 32-bit CRC

Algorithm	Data (DATA)
CRC-32	[31:0] = Valid data
CRC-32_BZIP2	[31:0] = Valid data
CRC-32C	[31:0] = Valid data
CRC-32D	[31:0] = Valid data
CRC-32_MPEG-2	[31:0] = Valid data
CRC-32_POSIX	[31:0] = Valid data
CRC-32Q	[31:0] = Valid data
CRC-32_JAMCRC	[31:0] = Valid data
CRC-32_XFER	[31:0] = Valid data

40.7 Memory map and register descriptions

NOTE

You must reconfigure CRC engine in case an IPS transfer error occurs (ips_xfr_err).

The CRC module generates a transfer error in the following cases:

- Write accesses to the register addresses that are not mapped to the peripherals but included in the address spaces of the peripherals.

40.7.1 CRC register descriptions

40.7.1.1 CRC memory map

CRC0 base address: 400C_B000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Data (DATA)	32	RW	FFFF_FFFFh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
4h	Polynomial (GPOLY)	32	RW	0000_1021h
8h	Control (CTRL)	32	RW	0000_0000h

40.7.1.2 Data (DATA)

Offset

Register	Offset
DATA	0h

Function

Configures the value of seed, data, and checksum. When CTRL[WAS] = 1, any write to this register is regarded as the seed value. When CTRL[WAS] becomes 0, any write to this register is regarded as data for general CRC calculation.

In 16-bit CRC mode, DATA[HU] and DATA[HL] are not used for programming the seed value, and reads of these fields return an indeterminate value. In 32-bit CRC mode, all fields are used for programming the seed value.

When programming data values, you can program to write 8 bits, 16 bits, or 32 bits in big endian order, provided all bytes are contiguous.

After writing all data values, you can read the CRC result from DATA register. In 16-bit CRC mode, the CRC result is available in DATA[LU] and DATA[LL]. In 32-bit CRC mode, all fields contain the result. After writing all data, you must wait for at least two clock cycles to read the data from CRC data (DATA) register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HU								HL							
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LU								LL							
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-24	Upper Part of High Byte

Table continues on the next page...

Table continued from the previous page...

Field	Function
HU	Generates CRC checksum in both 16-bit and 32-bit CRC modes if CTRL[WAS] = 0. • In 16-bit CRC mode (CTRL[TCRC] = 0), this field is not used for programming a seed value. • In 32-bit CRC mode (CTRL[TCRC] = 1), the values written to this field are part of the seed value when CTRL[WAS] = 1.
23-16 HL	Lower Part of High Byte Generates CRC checksum in both 16-bit and 32-bit CRC modes if CTRL[WAS] = 1. • In 16-bit CRC mode (CTRL[TCRC] = 0), this field is not used for programming a seed value. • In 32-bit CRC mode (CTRL[TCRC] = 1), the values written to this field are part of the seed value when CTRL[WAS] = 1.
15-8 LU	Upper Part of Low Byte Generates CRC checksum when CTRL[WAS] = 0. When CTRL[WAS] = 1,the values written to this field are part of the seed value.
7-0 LL	Lower Part of Low Byte Generates CRC checksum when CTRL[WAS] = 0. When CTRL[WAS] = 0, the values written to this field are part of the seed value.

40.7.1.3 Polynomial (GPOLY)

Offset

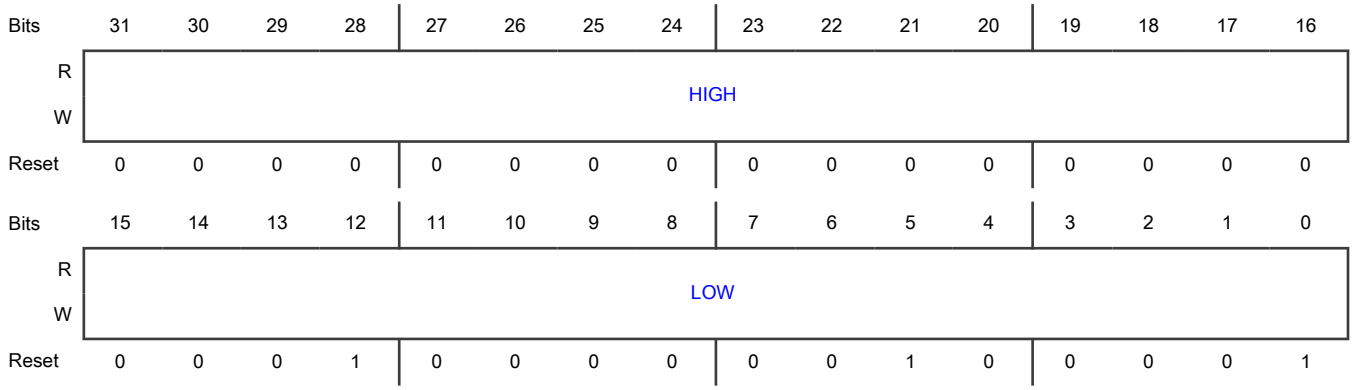
Register	Offset
GPOLY	4h

Function

Configures polynomial value for CRC calculation.

- Sets the upper 16 bits of polynomial that are used only in 32-bit CRC mode. Writes to this field are ignored in 16-bit CRC mode.
- Sets the lower 16 bits of polynomial that are used in both 16-bit and 32-bit CRC modes.

Diagram



Fields

Field	Function
31-16 HIGH	High Half-Word Writable and readable in 32-bit CRC mode (CTRL[TCRC] = 1). You cannot write to this field in 16-bit CRC mode (CTRL[TCRC] = 0).
15-0 LOW	Low Half-Word Writable and readable in both 16-bit and 32-bit CRC modes.

40.7.1.4 Control (CTRL)

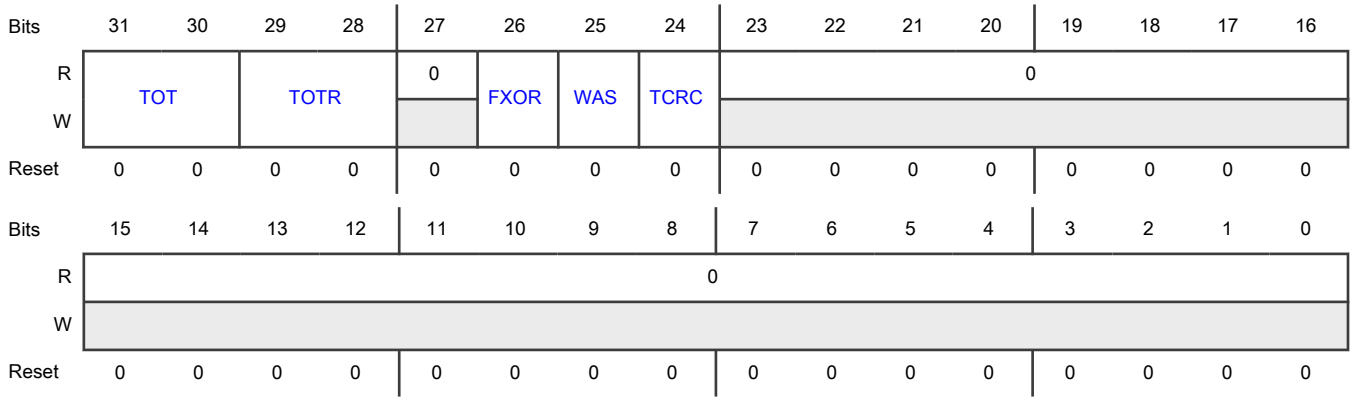
Offset

Register	Offset
CTRL	8h

Function

Sets control for CRC. You must write 1 to the appropriate fields of this register before starting a new CRC calculation, which you can initialize by writing 1 to CTRL[WAS] and then writing the seed into DATA.

Diagram



Fields

Field	Function
31-30 TOT	Transpose Type for Write Sets transpose type for the values written to DATA. See Transpose feature for the available transpose options. 00b - No transposition 01b - Bits in bytes are transposed, but bytes are not transposed. 10b - Both bits in bytes and bytes are transposed. 11b - Only bytes are transposed, no bits in a byte are transposed.
29-28 TOTR	Transpose Type for Read Sets transpose type for the values read from DATA . See Transpose feature for the available transpose options. 00b - No transposition 01b - Bits in bytes are transposed, but bytes are not transposed. 10b - Both bits in bytes and bytes are transposed. 11b - Only bytes are transposed, no bits in a byte are transposed.
27 —	Reserved
26 FXOR	Complement Read of CRC Data Register Enables on-the-fly complementing of read data. Some CRC protocols require the final checksum to be XORed with FFFFFFFFh or FFFFh. 0b - Disables XOR on reading data. 1b - Inverts or complements the read value of the CRC Data.
25 WAS	Write as Seed Specifies whether writes to DATA are data values or seed values. When this field = 1, the value that you write to is considered as seed value. When this field = 0, the value that you write to is considered as data for CRC calculation. 0b - Data values 1b - Seed values
24 TCRC	TCRC Defines the width of CRC. 0b - 16 bits 1b - 32 bits
23-0 —	Reserved

Chapter 41
Analog-to-Digital Converter (ADC)

41.1 Chip-specific ADC information

Table 357. Reference links to related information

Topic	Related module	Reference
Full description	ADC	ADC
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See ADC0_TRIG0-ADC0_TRIG3 and ADC1_TRIG0-ADC1_TRIG3 registers in INPUTMUX

41.1.1 Module instances

This device has two instances of 16-bit ADC, ADC0 and ADC1.

41.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

41.1.3 ADC input connections

Table 358. ADC analog channel input connections

ADC Channel (CMDLn[ADCH])	ADC0 A Connection	ADC0 B Connection	ADC1 A Connection	ADC1 B Connection	Input Type
0	ADC0_A0 (ANA_0)	ADC0_B0 (ANA_1)	ADC1_A0 (ANA_4)	ADC1_B0 (ANA_5)	High Speed
1	ADC0_A1/ OPAMP0_OUT (P4_15)	ADC0_B1/ OPAMP1_OUT (P4_19)	OPAMP0_INT	OPAMP1_INT	High Speed
2	ADC0_A2/ OPAMP2_OUT (P4_23)	ADC0_B2/ OPAMP2_OUT (P4_23)	OPAMP2_INT	OPAMP2_INT	High Speed
3	ADC0_A3/ DAC2_OUT (ANA_6)	OPAMP2_INT	ADC1_A3/ DAC2_OUT (ANA_6)	ADC1_B3/ OPAMP2_OUT (P4_23)	High Speed

Table continues on the next page...

Table 358. ADC analog channel input connections (continued)

4	ADC0_A4/ DAC0_OUT (P4_2)	ADC0_B4/ DAC1_OUT (P4_3)	ADC1_A4/ DAC0_OUT (P4_2)	ADC1_B4/ DAC1_OUT (P4_3)	Standard Dedicated
5	ADC0_A5/ OPAMP0_IN0P (P4_12)	ADC0_B5/ OPAMP0_IN1P (P4_13)	ADC1_A5/ OPAMP0_IN0P (P4_12)	ADC1_B5/ OPAMP0_IN1P (P4_13)	Standard Dedicated
6	ADC0_A6/ OPAMP1_IN0P (P4_16)	ADC0_B6/ OPAMP1_IN1P (P4_17)	ADC1_A6/ OPAMP2_IN0P (P4_20)	ADC1_B6/ OPAMP2_IN1P (P4_21)	Standard Dedicated
7	ADC0_A7/ VREFI/VREFO (ANA_7)	VREFL	ADC1_A7/ VREFI/VREFO (ANA_7)	VREFL	Standard Dedicated
8	ADC0_A8 (P0_16)	ADC0_B8 (P0_8)	ADC1_A8 (P1_8)	ADC1_B8 (P5_0)	Standard Muxed
9	ADC0_A9 (P0_17)	ADC0_B9 (P0_9)	ADC1_A9 (P1_9)	ADC1_B9 (P5_1)	Standard Muxed
10	ADC0_A10 (P0_18)	ADC0_B10 (P0_10)	ADC1_A10 (P1_10)	ADC1_B10 (P5_2)	Standard Muxed
11	ADC0_A11 (P0_19)	ADC0_B11 (P0_11)	ADC1_A11 (P1_11)	ADC1_B11 (P5_3)	Standard Muxed
12	ADC0_A12 (P0_20)	ADC0_B12 (P0_12)	ADC1_A12 (P1_12)	ADC1_B12 (P5_4)	Standard Muxed
13	ADC0_A13 (P0_21)	ADC0_B13 (P0_13)	ADC1_A13 (P1_13)	ADC1_B13 (P5_5)	Standard Muxed
14	ADC0_A14 (P0_22)	ADC0_B14 (P0_14)	ADC1_A14 (P1_14)	ADC1_B14 (P5_6)	Standard Muxed
15	ADC0_A15 (P0_23)	ADC0_B15 (P0_15)	ADC1_A15 (P1_15)	ADC1_B15 (P5_7)	Standard Muxed
16	ADC0_A16 (P1_0)	ADC0_B16 (P0_24)	ADC1_A16 (P1_16)	ADC1_B16 (P5_8)	Standard Muxed
17	ADC0_A17 (P1_1)	ADC0_B17 (P0_25)	ADC1_A17 (P1_17)	ADC1_B17 (P5_9)	Standard Muxed
18	ADC0_A18	ADC0_B18 (P0_26)	ADC1_A18 (P1_18)	—	Standard Muxed

Table continues on the next page...

Table 358. ADC analog channel input connections (continued)

	(P1_2)				
19	ADC0_A19 (P1_3)	ADC0_B19 (P0_27)	ADC1_A19 (P1_19)	—	Standard Muxed
20	ADC0_A20 (P1_4)	ADC0_B20 (P0_28)	ADC1_A20 (P1_20)	—	Standard Muxed
21	ADC0_A21 (P1_5)	ADC0_B21 (P0_29)	ADC1_A21 (P1_21)	—	Standard Muxed
22	ADC0_A22 (P1_6)	ADC0_B22 (P0_30)	ADC1_A22 (P1_22)	—	Standard Muxed
23	ADC0_A23 (P1_7)	ADC0_B23 (P0_31)	ADC1_A23 (P1_23)	—	Standard Muxed
24	OPAMP1_OBS	OPAMP0_OBS	OPAMP1_OBS	OPAMP0_OBS	Internal
25	VDD_CORE/4	OPAMP2_OBS	VDD_SYS/4	OPAMP2_OBS	Internal
26	Temperature+	Temperature-	Temperature+	Temperature-	Internal
27	PMC BG+	VSS	PMC BG+	VSS	Internal
28	VREF BG+	VREF BG-	VREF BG+	VREF BG-	Internal
29	VBAT/4	—	VDD_DCDC/4	—	Internal

41.1.4 ADC voltage reference options

The ADC voltage references are:

- CFG[REFSEL]=00, VREFH reference pin
- CFG[REFSEL]=01, ANA_7(VREFI/VREFO) pin
- CFG[REFSEL]=10, VDD_ANA supply pin

41.1.5 ADC trigger inputs

ADC trigger sources get routed through the Input Multiplexing (INPUTMUX). See the INPUTMUX chapter for available trigger sources.

41.1.6 ADC input clock frequency

See device datasheet to know minimum/maximum ADC input clock frequencies.

41.2 Overview

The 16-bit analog-to-digital converter (ADC) is a dual successive approximation ADC designed for operation within an integrated microcontroller system-on-a-chip.

NOTE

For chip-specific modes of operation, see the power management information for the device.

41.2.1 Block diagram

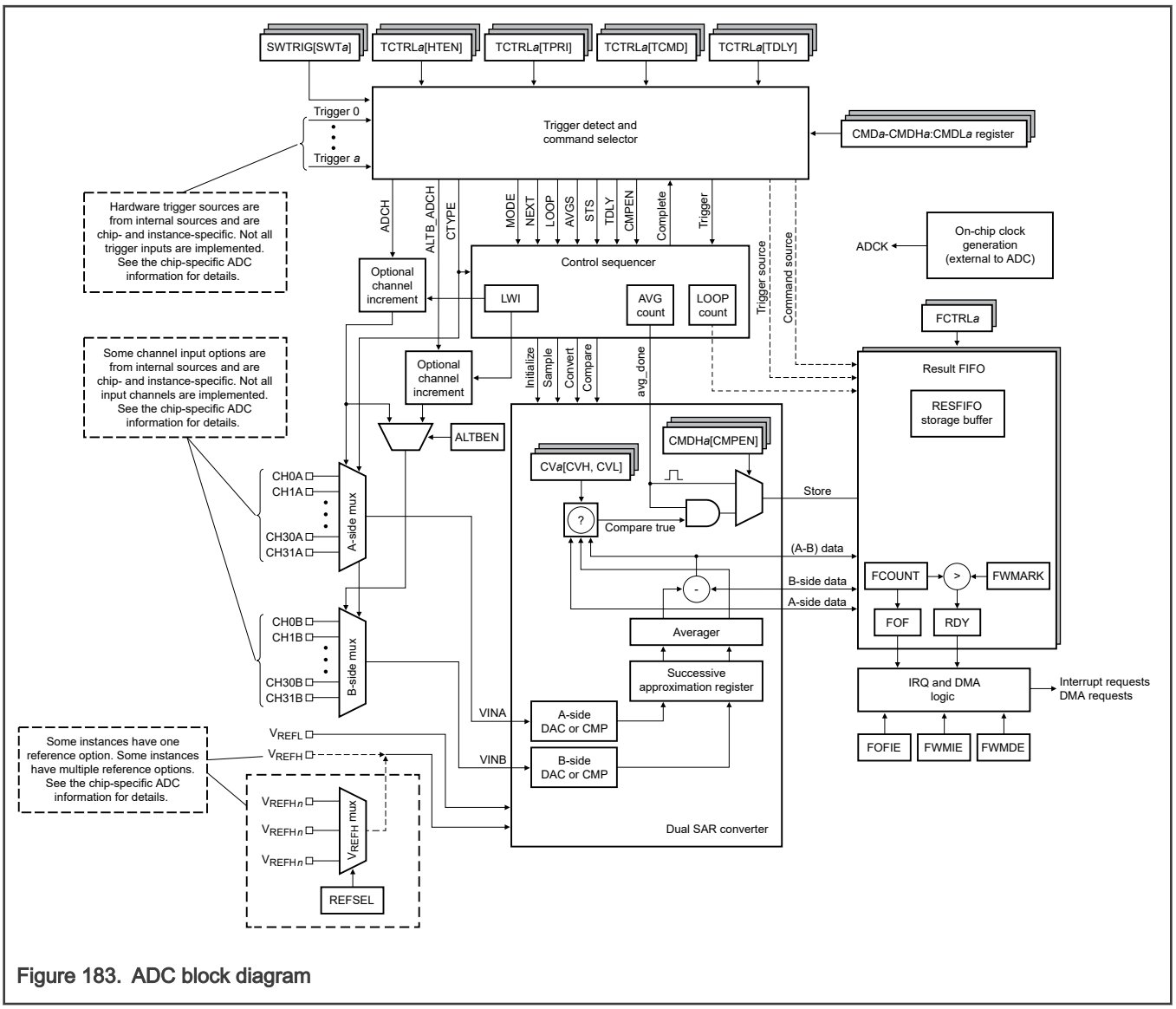


Figure 183. ADC block diagram

41.2.2 Features

ADC has these features:

- Linear successive approximation algorithm
  - Differential operation with 16-bit or 13-bit resolution
  - Single-ended operation with 16-bit or 12-bit resolution
  - Support for two simultaneous single-ended conversions
- Configurable analog input sample time
- Configurable speed options to accommodate operation in low-power modes of SoC
- Trigger detection with up to 4 trigger sources with priority level configuration. Software or hardware trigger option for each.
- 15 command buffers, to allow independent options selection and channel sequence scanning

- Automatic comparisons for less-than, greater-than, within range, or out-of-range with "store on true" and "repeat until true" options
- 2 independent result FIFOs, each containing 16 entries. Each FIFO has configurable watermark and overflow detection.
- Interrupt, DMA, or polled operation
- Linearity and gain adjustment calibration logic

41.3 Functional description

ADC performs analog-to-digital conversions on any of the software-selectable analog input channels via a successive approximation algorithm.

The ADC module can average the result of multiple conversions on a channel before storing the calculated result. The hardware average function is enabled by setting CMDHn[AVGS] to a non-zero value. The function operates in any conversion mode or configuration.

ADC can compare the result of a conversion with the contents of two value registers for less-than, greater-than, inside-range, or outside-range detection. The compare function operates in any conversion mode or configuration.

When the conversion and averaging loops finish, the resulting data is placed in one of 2 available FIFO data buffers . The data includes tag information associated with the result. When the number of stored data words exceeds the setting, a configurable watermark level supports interrupts or DMA requests. Interrupts can also be enabled to indicate when FIFO overflow errors occur.

The module initializes to its lowest power state during reset.

The ADC analog circuits can be pre-enabled to begin conversions sooner at the expense of higher idle currents. Conversions are initiated by selectable trigger events from software or hardware sources.

The trigger-detection logic includes a configurable enable and priority scheme for available trigger sources.

ADC includes multiple command buffers to provide flexibility for channel scanning and independent channel selections for different trigger sources. These command buffers can be configured for:

- Differential or single-ended conversion
- Sample time
- Averaging on a per-channel basis

ADC includes offset and linearity calibration logic. A request for calibration should be made upon reset or power up. Each successive approximation register (SAR) conversion uses calibration data calculated during the calibration routine.

The sequencing of an ADC command is summarized in the flow diagram below.

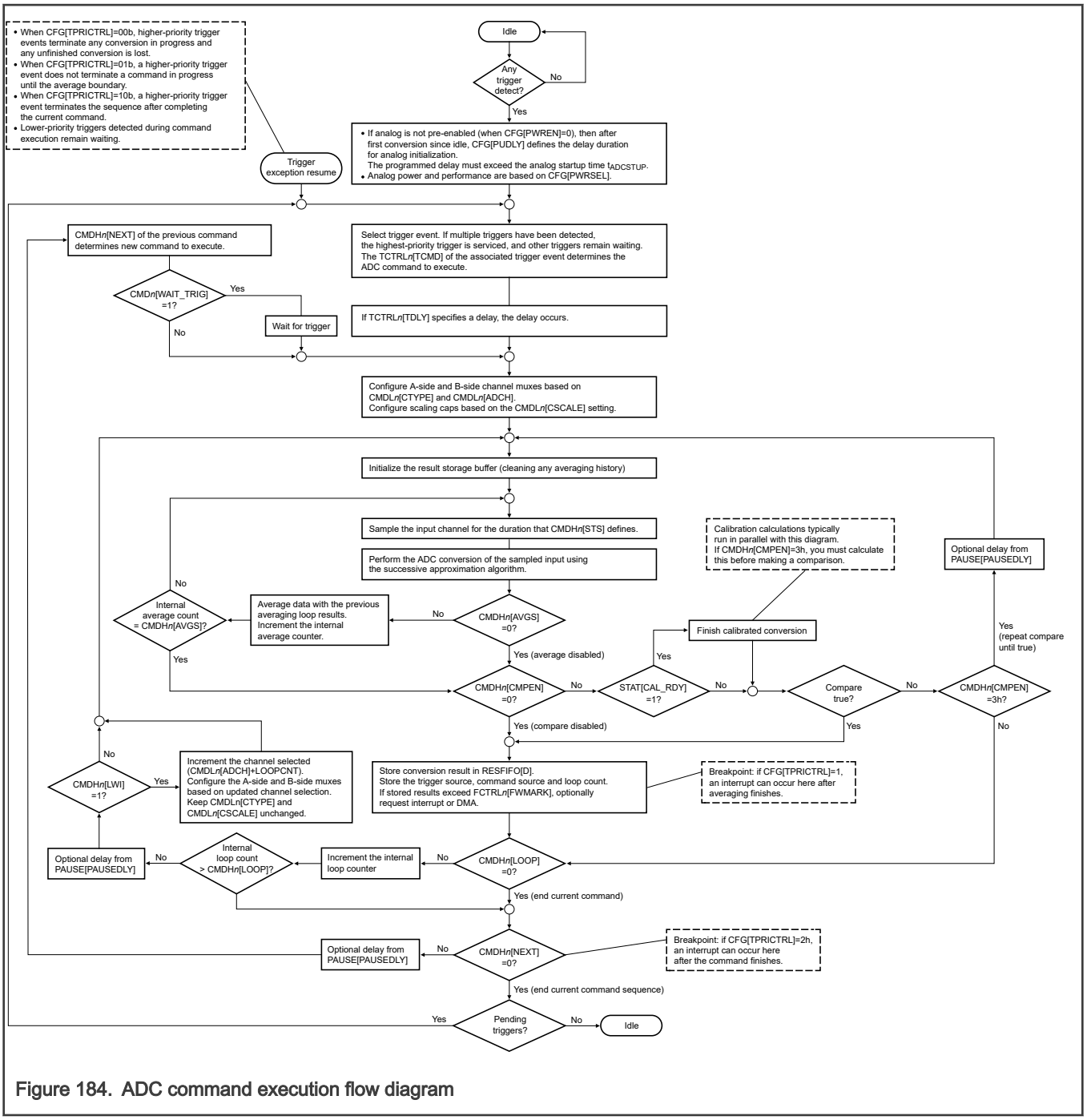


Figure 184. ADC command execution flow diagram

41.3.1 Power control mode

By default, the ADC analog circuits are disabled while ADC is in its idle state. When a trigger is detected and ADC command processing is initiated, the analog circuits are enabled. These circuits require a period of initialization before the first conversion cycle.

The value of CFG[PUDLY] should be set to incur a delay longer than $t_{ADCSTUP}$. The accuracy of initial conversions after activation is degraded if the value of CFG[PUDLY] is too low.

You can achieve faster conversion startup times by setting CFG[PWREN] to pre-enable the analog circuits of ADC. This faster conversion consumes extra power, even while ADC is in an idle state. When CFG[PWREN] is 1, the Power Enable timer is

activated. The timer enforces the minimum time required (configured by CFG[PUDLY]) before detected triggers can initiate ADC conversions.

ADC also has options for controlling power and performance summarized in the table below. See the device data sheet for specification of the available power modes.

Table 359. Power option settings

CFG[PWRSEL]	Description
0xb (Default)	Slow speed and low power
1xb	Fast speed and high power

41.3.2 ADC behavior in low-power mode

ADC supports system-level low-power modes. In low-power modes where ADC can remain functional, CTRL[DOZEN] controls ADC behavior.

When CTRL[DOZEN] = 0, if ADC was configured and enabled before entering low-power mode, ADC can continue operating while the system transitions to the low-power state. Any conversion in progress is not disrupted. External hardware trigger detection and active conversions are operational.

When CTRL[DOZEN] = 1, ADC waits for the current averaging iteration or FIFO storage to complete before acknowledging the low-power entry request.

When entering a low-power mode where ADC cannot remain functional, the module:

- Ignores CTRL[DOZEN].
- Waits for the current averaging iteration or FIFO storage to complete before acknowledging the low-power entry request.

After entering the low-power state, the ADC is kept inactive until the system exits the low-power state. The setting of CFG[PWREN] is ignored and the ADC analog circuits are forced to their lowest-power state. Upon waking from the low-power state, the analog circuits are automatically re-enabled and the power-up delay timer begins, if:

- The analog circuits were pre-enabled before entering Stop mode (CFG[PWREN] = 1), and
- The control registers retained their states in low-power mode.

Any triggered conversion is stalled until the power-up delay associated with CFG[PUDLY] has finished.

41.3.3 Voltage reference

The voltage reference high (V_{REFH}) used by ADC is supplied from either an on-chip voltage reference source or from an off-chip source via external pins. V_{REFL} is always from an external pin and must be at the same voltage as the Analog Power Domain Ground.

This block supports a programmable selection of the Voltage Reference High used for ADC conversions (via CFG[REFSEL]).

See the chip configuration information on the voltage reference options specific to this packaged device.

41.3.4 Trigger detect and command execution

See Figure 184 for a flow diagram of command execution sequencing.

ADC command execution is initiated from up to 4 trigger sources. Each trigger can be software-generated by writing 1b to the corresponding SWTRIG[SWTn] field. Alternatively, asynchronous input sources at the periphery of the module can generate hardware triggers. The number and sources of hardware triggers implemented is device-specific. See the chip-specific ADC information for descriptions of available hardware trigger sources for this device.

Each hardware trigger source is enabled by setting the associated enable field (TCTRLn[HTEN]). Each trigger source is assigned a priority via the associated priority control field (TCTRLn[TPRI]). Each of the trigger sources is associated with a command buffer via the associated command select field (TCTRLn[TCMD]).

When a hardware trigger input is enabled, hardware trigger events are detected on the rising edge of the associated hardware trigger source.

Each trigger source has an associated priority field TCTRLn[TPRI] which allows arbitration between trigger sources. Arbitration selects two things: which trigger sequence to execute next, and how to handle a trigger exception. Trigger exceptions are defined as allowing a higher-priority trigger sequence to interrupt operation of a lower-priority sequence. When a trigger exception occurs, programmable arbitration allows the configurable stop and resume points for low-priority sequences. The fields affecting arbitration are CFG[HPT_EXDI], CFG[TCMDRES], CFG[TRES], and CFG[TPRICTRL].

1. When CFG[HPT_EXDI] = 1b, trigger exceptions are disabled and any higher priority triggers are left pending until the current sequence completes. New triggers are accepted based on priority.
2. When CFG[HPT_EXDI] = 0b (default), exceptions are enabled and the higher-priority sequence begins executing at a user-specified breakpoint.

Breakpoint locations are determined by CFG[TPRICTRL], which affects latency for accepting trigger exceptions.

1. When CFG[TPRICTRL] = 0h, a higher-priority trigger causes an immediate command abort and the new command specified by the trigger is immediately started.
2. When CFG[TPRICTRL] = 1h, the current conversion is allowed to complete (including averaging) before the higher-priority exception starts. In this mode, if the command is running through a series of averages, this series completes. However, there is no requirement to finish the entire command before being interrupted. For example, if the command consists of four loop iterations, there is no requirement to complete all four iterations before the interrupt occurs.
3. When CFG[TPRICTRL] = 2h, a higher-priority trigger begins once the current command is completed. If a command consists of five loop iterations each containing eight averages, then all 40 conversions must be completed before accepting the trigger exception.

CFG[TCMDRES] and CFG[TRES] determine what ADC does after accepting a trigger exception. The module can be programmed to resume commands after returning from a trigger exception.

1. When CFG[TRES] = 0h, commands are not automatically resumed after being stopped by an exception. However, an interrupt is set to indicate that this case has occurred. The flag TSTAT[TEXC_NUM] can be used to resolve which trigger was stopped by the exception.
2. When CFG[TRES] = 1h, ADC automatically resumes commands after they were stopped by an exception.

Using CFG[TRES] with CFG[TCMDRES], the module can be programmed to resume commands at one of two possible locations.

1. When CFG[TCMDRES] = 0h, the trigger which was stopped by an exception is resumed from the beginning of its associated command sequence. Triggers that are waiting to be resumed take the same priority programmed to TCTRLn[TPRI].
2. When CFG[TCMDRES] = 1h, the trigger is resumed from the command that it was executing before being interrupted by an exception.

If a trigger occurs with priority lower than the priority of the currently executing command, trigger detection is left pending until the current command sequence finishes. Lower-priority trigger events cannot be serviced until a higher-priority triggered command (or command sequence) completes.

When a conversion is completed (including hardware averaging when CMDHn[AVGS] is non-zero), the result is placed in a RESFIFO buffer. When an ADC command selects looping (when CMDHn[LOOP] is non-zero) a command stores multiple conversion results to the FIFO during execution of that command.

At the end of command execution, CMDHn[NEXT] selects the next command to be executed. Multiple commands can be executed sequentially by configuring the CMDHn[NEXT] field of each command. Setting the next command to 0h causes conversions to terminate at the completion of the current command. Unending circular command execution is allowed by setting the CMDHn[NEXT] field of the last command in a sequence to the first command in the sequence.

By default, command sequences execute automatically in the order in which CMDHn[NEXT] fields are programmed. However, by using CMDHn[WAIT_TRIG], command execution can be stalled and launched based on trigger inputs. For example, if TRIGGER2 is programmed to start the command sequence CMD1, CMD2, CMD3, then this sequence is run once unconditionally to

completion upon receiving TRIGGER2. If CMDH2[WAIT_TRIG] = 1h, however, the sequence pauses after CMD1 until TRIGGER2 is received again. In this way, sequences can be stalled until a trigger assertion is received.

Disabling ADC by writing 0b to CTRL[ADCEN] terminates any active ADC command processing. Writing 0b to CTRL[ADCEN] causes the current command (or command sequence) to terminate, clears any pending triggers, and sends the ADC module to an idle state.

41.3.5 Pause option

When an application does not require the maximum conversion rate, the effective conversion rate can be reduced:

- By implementing periodic trigger events to initiate ADC conversions, or
- By selecting a reduced-frequency clock as the ADCK source.

Both options are chip-specific and are dependent on ADC triggering and clocking options external to the ADC module. The latency associated with ADC analog power-up delays limits the maximum conversion rate when using periodic triggering.

Conversion rates can also be reduced by inserting a pause of a programmable duration:

- Between loop iterations
- Between commands in a sequence
- Between conversions, when a command is executing in the Compare Until True configuration

When PAUSE[PAUSEEN] = 1, PAUSE[PAUSEDLY] controls the duration of pausing during command execution sequencing. The pause delay is a count of (PAUSE[PAUSEDLY] * 4) ADCK cycles.

NOTE

Do not change the PAUSE register while CTRL[ADCEN] = 1. Writes to the PAUSE register while CTRL[ADCEN] is 1 can lead to metastable operation.

See Figure 184 for the places during command execution sequencing where the pause can be inserted.

41.3.6 Resync functionality

Any software or hardware trigger source can be configured to act as a resync trigger. Trigger-based resync interrupts a running trigger (resync target) and clears the FIFO to which the trigger is writing. Resync can be used either to abort a running sequence, or to restart a running sequence depending on the value of CFG[TRES]. If CFG[TRES] = 1, the target sequence is aborted, the FIFO is cleared, and the sequence is restarted after the resync occurs. If CFG[TRES] = 0, the target sequence is aborted and the FIFO is cleared after the resync occurs. The FIFOs that are cleared are based on the resync targets TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B]. To only clear one FIFO, set TCTRLn[FIFO_SEL_A] = TCTRLn[FIFO_SEL_B]. When the resync occurs, any results are stored in TCTRLn[FIFO_SEL_A] or TCTRLn[FIFO_SEL_B] that are not associated with the resync target are lost.

A resync trigger must have a specific target. The resync only occurs if the resync target is running at the time of the trigger. In the following example, n is the resync trigger number and m is the resync target number. According to these variables, trigger n should resync trigger m. To enable a trigger source to act as a resync trigger, these conditions must be satisfied:

1. The resync trigger TCTRLn[RSYNC] must be set to 1.
2. The resync trigger must have higher priority than the resync target (TCTRLn[TPRI] must be less than TCTRLm[TPRI]).
3. The resync target is specified using TCTRLn[TCMD]. In this case the resync target, m, must be equal to TCTRLn[TCMD].
4. The resync target, m, must be executing commands when the resync trigger, n, is asserted.
5. Trigger m must have at least one conversion remaining to start when trigger n is received.

If a trigger source n has TCTRLn[RSYNC] set to 1, but some of the above conditions are not met, the trigger source n is ignored.

The following figure illustrates a resync trigger sequence. In this example, trigger source 1 is configured to resync trigger source 0.

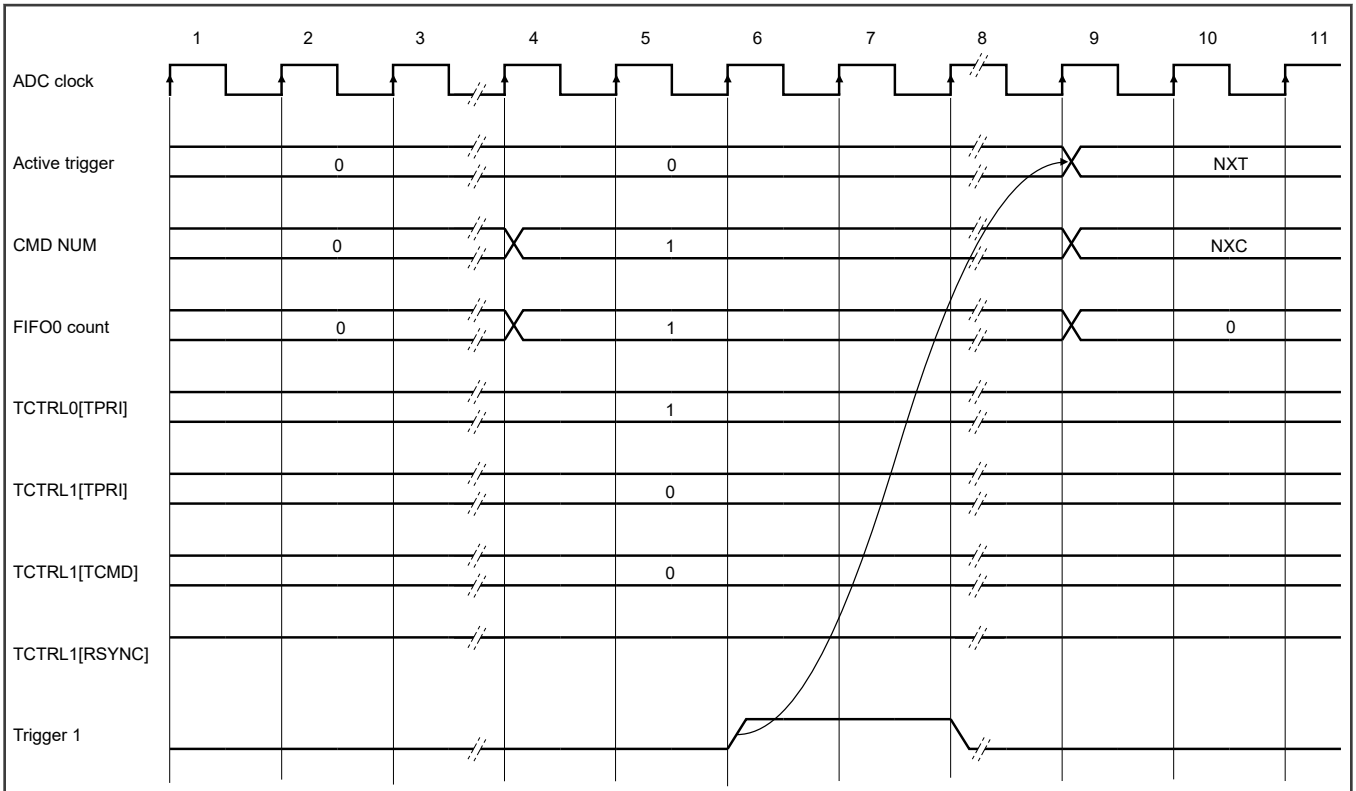


Figure 185. ADC Resync Example

In this figure, trigger source 0 is executing a sequence of commands when trigger source 1 is asserted. Trigger source 0 is stopped when trigger source 1 is asserted after some synchronization delay. In addition, the FIFO being written to by trigger source 0 is cleared (FIFO0 in this example). After the trigger 0 sequence is stopped, ADC runs the next trigger pending with the highest priority. This trigger is marked as NXT, for next trigger. If resume functionality is enabled, and trigger source 0 has the highest priority pending, then NXT = 00h.

41.3.7 Temperature sensor

ADC has a dedicated input channel for an on-chip temperature sensor. See the chip-specific channel definition to determine which channel is connected to the on-chip temperature sensor.

To calculate the temperature, you must first execute a conversion of the temperature sensor channel with configuration requirements. The sequence to convert the temperature sensor channel is:

1. Configure a command buffer register to convert the temperature sensor channel (CMDLn[ADCH]).
2. Program the command buffer with the following parameters:
   - 16b mode (CMDLa[MODE] = 0x1)
   - Differential: CMDLn[CTYPE] = 2h
   - Max averaging: CMDHn[AVGS] = 7h
   - Max sample time: CMDHn[STS] = 7h
   - LOOP set to 1: CMDHn[LOOP] = 1h
   - Loop with increment disabled: CMDHn[LWI] = 0h
   - Compare function disabled: CMDHn[CMPEN] = 0h
3. Configure a trigger control register TCTRLn with the following parameters:

- TCTRLn[TCMD] = command buffer used in steps 2 and 3.
- TCTRLn[FIFO_SEL_A] directs conversion results to FIFO0 orFIFO1

4. Trigger a conversion of the temperature sensor channel. You trigger the conversion by writing 1 to the associated bit in Software Trigger Register (SWTRIG) (that is, the trigger configured in the previous step).

After completing the conversion of the temperature sensor channel, two results are stored in the FIFO selected by TCTRLn[FIFO_SEL_A]. Each result corresponds to a component of the overall temperature value. Read these values from the FIFO and use the following equation to calculate the ambient temperature.

$$\text{Temp} = A\left[\frac{\alpha(\text{Vbe8}-\text{Vbe1})}{\text{Vbe8} + (\alpha(\text{Vbe8}-\text{Vbe1}))}\right] - B$$

Equation 17. Temperature sensor function

In the preceding equation:

- Vbe1 is the first value stored to the FIFO as a result of the temperature sensor channel conversion.
- Vbe8 is the second value stored to the FIFO as a result of the temperature sensor channel conversion.
- A is the slope factor.
- B is the offset factor.
- α is the bandgap coefficient.

A, B, and α are specified constant values from the ADC electrical information in the chip data sheet.

41.3.8 Result FIFO operation

ADC includes 2 16-entry FIFOs in which the result of ADC conversions are stored. In addition, a valid indicator bit, the trigger source, the source command, and the loop count are also stored with the data. FCTRLn[FCOUNT] indicates how many valid data words are stored in each RESFIFO.

A programmable watermark threshold supports configurable notification of data availability. When FCTRLn[FCOUNT] is greater than FCTRLn[FWMARK], the associated RDY flag is asserted. When IE[FWMIEn] = 1, a watermark interrupt request is issued. When DE[FWMDEn] = 1, a DMA request is issued. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements FCTRLn[FCOUNT]. When FCTRLn[FCOUNT] falls equal to or below FCTRLn[FWMARK], the RDY flag is cleared.

Each FIFO can be emptied by successive reads of RESFIFOn. When RESFIFOn[VALID] is 1, the associated FIFO entry is valid. Reading RESFIFOn when the FIFO is empty (when RESFIFOn[VALID] = 0 and FCTRLn[FCOUNT] = 0h) provides an undefined data word. All FIFOs are reset by writing 1b to CTRL[RSTFIFOn].

If ADC attempts to store a data word to the FIFO when the FIFO is full, the FIFO overflow flag (FCTRLn[FOF]) is set. When IE[FOFIEn] = 1, an overflow interrupt request is issued. The FOF flag is cleared by writing 1 to STAT[FOFn]. When overflow events occur, no new data is stored and the data associated with the storage event that triggered the overflow is lost.

Conversion results can be steered to any FIFO in the design. TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B] determine into which FIFO the final result is written. Depending on which trigger is executing, the results can be steered to different locations. Depending on the type of conversion selected, the FIFO destination register fields are interpreted differently. During either differential or single-ended mode (CMDLn[CTYPE] != 3h) only one result is produced. The destination during these modes is determined from TCTRLn[FIFO_SEL_A]. In dual-single-ended mode, both TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B] determine the Channel A and Channel B destinations respectively.

41.3.9 Sampling Modes

The ADC module supports three different sampling modes: dual single-ended, single-ended, and differential. The sampling mode is determined by the currently executing command using CMDLn[CTYPE].

When executing a command in dual single-ended mode, two independent conversion results are calculated and stored in selectable FIFO destinations. Command processing, however, is not individually controlled for each independent channel being sampled. When operating in dual single-ended mode both channels are sampled and processed simultaneously.

The selection of Channel B is controlled by CMDLn[ALTBEN].

- When CMDLn[ALTBEN] = 1, the channel for the B-side is selected independently (via the value of CMDLn[ALTB_ADCH]).
- When CMDLn[ALTBEN] = 0, the Channel B selection is controlled by CMDLn[ADCH] and Channel A and Channel B are paired (CH0A/CH0B, CH1A/CH1B, and so on).

If comparisons are enabled in dual single-ended mode, only the A-side channels (CH0A, CH1A, and so on) are used for the comparison. The A-side comparison determines whether both the A-side and B-side results are written to the FIFOs.

- If the A-side comparison passes, both the A-side and B-side results are stored.
- If the A-side comparison fails, A-side and B-side results are not written to the FIFOs.

Single-ended mode is configurable to allow either A-side or B-side channels to be sampled. In single-ended mode, the results from each conversion can be written to a selectable FIFO using TCTRLn[FIFO_SEL_A].

In differential mode, the final SAR calculation is equivalent to V(CHA) - V(CHB). If the result is negative, then the value is stored in sign-extended two's complement format. TCTRLn[FIFO_SEL_A] also determines where differential conversion results are stored.

In dual single-ended mode, however, two independent results are produced. Individual control is provided by using TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B] to select a FIFO destination for both results during this mode. Both single-ended results may be written to the same destination by programming TCTRLn[FIFO_SEL_A] = TCTRLn[FIFO_SEL_B]. In this case, the CH_A result is always stored before the CH_B result.

41.3.10 Compare Function

After the input is sampled and converted and any averaging iterations are performed, CMDHn[CMPEN] determines:

- Whether to use the automatic compare function, storing the conversion result when true.
- Whether to repeat the channel acquisition until the automatic compare function returns a true result.

The command-sequencing options related to the compare function are summarized in the table below.

NOTE

Latency is added to the end of a compare-until-true conversion to resolve the next command or loop in a sequence. This latency is necessary to calibrate the SAR data before resolving the result of a comparison. Delay for this feature is only added when resolving the result of a conversion. Intermediate samples during averaging do not include extra latency; only loop and command boundaries experience this delay. The latency is always less than or equal to five ADC clock cycles.

Table 360. Compare modes

CMDHn[CMPEN]	Compare function	Description
00b	Compare disabled	Do not perform compare operation. Always store the conversion result to the FIFO.
01b	Reserved	
10b	Store on true	Perform compare operation. Store conversion result to FIFO after averaging only when the result of the comparison is true. Regardless of the comparison result, the LOOP setting is considered. The LOOP counter is incremented before deciding whether the current command has completed or additional LOOP iterations are required.
11b	Repeat compare until true	Perform compare operation. Store conversion result to FIFO after averaging only when the result of the comparison is true. Once a comparison is true, the LOOP setting is considered. The LOOP counter is incremented before deciding whether the current command has completed or additional LOOP iterations are required. When a

Table continues on the next page...

Table 360. Compare modes (continued)

CMDHn[CMPEN]	Compare function	Description
		comparison is false, the conversion is repeated without considering the LOOP setting, and the LOOP counter is not incremented.

The compare operation checks the result based on the values of CVn[CVH] and CVn[CVL], as shown in the following table.

Table 361. Compare operations

CVn[CVL] vs. CVn[CVH]	Operation	Description
set CVn[CVL] < CVn[CVH]	Outside range (General form)	True if the result is less than CVn[CVL] **or** greater than CVn[CVH].
set CVn[CVH] to maximum value set CVn[CVL] to compare point	Less than	True if the result is less than CVn[CVL].
set CVn[CVL] to minimum value set CVn[CVH] to compare point	Greater than	True if the result is greater than CVn[CVH].
set CVn[CVL] > CVn[CVH]	Inside range	True if the result is less than CVn[CVL] **and** greater than CVn[CVH].

NOTE

When ADC continues operating in a low-power mode, the compare function can monitor the voltage and wake the device only when the compare condition is met.

41.3.11 Cycles per conversion

The number of ADCK cycles needed to complete a conversion, varies based on the selected resolution (CMDLa[MODE]), the sample time configured, and several other configuration options discussed below.

To calculate the cycle count, first, determine the Base Cycles Count from Table 362. Next, add the Sample Time Adder based on CMDHa[STS] setting as summarized in Table 363. If averaging is enabled (CMDHa[AVGS] does not equal to 0), then the total cycle (Base Cycles Count + Sample Time Adder) should be multiplied by the number of averages configured as summarized in Table 364.

$$CycleCount\ /\ Conversion = [(BaseCycleCount + SampleTimeAdder) * AverageMultiplier]$$

Equation 18. Cycle Count/Conversion

Note that there is latency associated with trigger detection and starting an initial conversion. There is additional latency associated with offset and gain error adjustment of a raw conversion result and storage of a final result to the FIFO. Due to the pipe-lining of conversions when looping or command chaining is configured, the next conversion is immediately started while a raw conversion result is being adjusted and stored and thus the conversion rate is maximized.

The base cycles per conversion are variable depending on CMDLa[MODE] setting as summarized in the following table:

Table 362. Base cycles per conversion

Base ADCK cycles/conversion[1]	
16-bit CMDLa[MODE] = 1	12-bit CMDLa[MODE] = 0
24	19

1. This cycle count includes min sample time of 3.5 ADCK cycles.

In addition to the base cycles per conversion, the configured sample time needs to be considered. The sample time adder is variable depending on CMDHa[STS] setting and is summarized in the following table:

Table 363. Sample time cycle adder

CMDHa[STS] (default 000)	Add ADCK cycles/conversion
000	0
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Table 364. Averaging multiplier

CMDHa[AVGS] (default 0000)	Averaging multiplier
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1010	512
1011	1024

When LOOPing is used:

CMDHa[LOOP] allows iteration of a command (that is a channel can be converted multiple times). This feature adds 3 additional cycles between conversions.

NOTE

When commands are chained using CMDHa[NEXT] to execute back-to-back conversions the ADC does not have the 3 stall cycles between commands.

For a command that executes multiple conversions of the same channel using LOOPing the total cycle count is:

$$CycleCount/Command = [(BaseCycleCount + SampleTimeAdder) * AverageMultiplier * (CMDHa[LOOP] + 1)] + (3 * CMDHa[LOOP])$$

Equation 19. Cycle Count/Command

41.3.12 Clocking

ADC uses the ADCK clock input provided by an on-chip clock select block. It is used by the SAR conversion-control sequencing logic and the FIFO storage buffer. The ADCK frequency must be within the specified frequency range for ADCK, and varies based on CFG[PWRSEL]. See the device data sheet for supported frequency ranges.

ADC continues operating in low-power mode as long as CTRL[DOZEN] = 0 and the on-chip clock select block supplies an ADCK clock source.

NOTE

When in low-power mode with CTRL[DOZEN] = 0, the bus clock can be shut off, and asynchronous interrupts and DMA requests can be configured. ADC continues processing commands and writing data to the internal FIFO.

ADC has four sources for asynchronous interrupts during low-power mode: watermark, FIFO overflow, TCOMP, and TEXC. To enable them, configure IE[FWMIEx], IE[FOFIEx], IE[TCOMP_IE], and IE[TEXC_IE] before entering low-power mode.

When CTRL[DOZEN] = 1 in low-power mode, ADC waits for the current averaging iteration or FIFO storage to complete before acknowledging the low-power mode entry. Any pending triggers are dropped when a low-power mode request is made in this mode. ADC is forced into its lowest-power setting after acknowledging the low-power mode request.

41.3.13 Resets

Table 365. ADC Resets

Reset source	Description
Chip reset	The logic and registers for ADC are reset to their default states on a chip reset.
Software reset	ADC implements a software reset field in its control register. CTRL[RST] resets all logic and registers to their default states, except for the CTRL register itself.
FIFO reset	ADC implements write-only control that resets FIFO0 (CTRL[RSTFIFO0]) and FIFO1 (CTRL[RSTFIFO1]). After a FIFO is reset, that FIFO is empty.

41.3.14 Interrupts and DMA requests

ADC includes several sources for interrupts and DMA requests. The table below summarizes these sources.

A programmable watermark threshold supports configurable notification of data availability and can generate either an interrupt exception or a DMA request. When FCTRLn[COUNT] is greater than FCTRLn[FWMARK], the associated STAT[RDYn] flag is asserted. The IE[FWMIEn] and DE[FWMDEn] control fields control the masking for this exception. When STAT[RDYn] becomes 1 and IE[FWMIEn] = 1, a watermark interrupt request is issued. When STAT[RDYn] becomes 1 and DE[FWMDEn] = 1, a DMA request is issued.

The other exception sources can only generate interrupts and do not have a DMA request option. Each source has a mask control field in the IE register and no corresponding field in the DE register.

Table 366. ADC Interrupts and DMA Requests

Status Register (STAT)		Description	Can generate		
Flag	Name		Interrupt?	DMA request?	Low-power wake-up?
RDYn	Result FIFO n Ready Flag	Conversion result data is written to Result FIFO and has a watermark configurable trigger level to generate an exception request as controlled by FCTRLn[FWMARK].	Y	Y	Y
FOFn	Result FIFO n Overflow Flag	Attempting to store data to the FIFO when the FIFO is full is an error condition.	Y	N	Y
TCOMP_INT	Trigger Completion Flag	A trigger sequence has been completed. All associated commands have been run.	Y	N	Y
TEXC_INT	High Priority Trigger Flag	A high priority trigger exception has occurred.	Y	N	Y

41.4 External signals

The ADC module supports analog channel inputs with differential and single-ended conversion options for all channels.

See chip-specific section for supply and ground connections.

Table 367. ADC signal descriptions

Signal	Description	I/O
CHnA - CH0A[1]	A-side Analog Channel Inputs	I
CHnB - CH0B[1]	B-side Analog Channel Inputs	I

1. where n is the maximum channel number supported in the chip. See the chip-specific information for the number of channels supported on your device.

41.4.1 Analog channel inputs (CHnA and CHnB)

CMDLn[ADCH] and CMDLn[CTYPE] control selection of paired or individual input channels. Each ADC command independently makes a channel and conversion type selection. Each CMDLn[ADCH] channel selection has an associated A side and an associated B side input. Each CMDLn[ADCH] pair can be converted in differential mode, but only limited pairs should be converted as differential channels (for example, adjacent pins designed with matched impedance). For the pin pairings available for differential conversions for your device, see the Chip Configuration details.

NOTE

Some input channels from on-chip sources, such as temperature sensors and reference voltage sources, may only be connected to individual instances of ADC. Some input channel options in the field-setting descriptions may not be available for your device. See chip-specific information for the channels supported on this device.

41.5 Initialization

41.5.1 Calibration functions

The ADC module has multiple calibration functions that must be executed as part of ADC setup to achieve the specified accuracy. Calibration must be run after any reset and before a conversion is initiated. Before offset calibration or calibration, the user must configure the clock source and frequency of ADC for the clock source availability and needs of the application. ADC must be

enabled (CTRL[ADCEN] = 1h) before a calibration function runs. If calibration is requested while ADC is actively converting, that sequence completes before starting calibration.

Averaging multiple conversions can achieve improved accuracy during the calibration routines. CTRL[CAL_AVGS] is used during a calibration routine to control how many samples are averaged together. If the application uses ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected. Alternatively, multiple calibrations can be done for the different configurations.

41.5.1.1 Offset calibration

The Offset Trim Register (OFSTRIM) is used to trim for ADC comparator offset voltage. The ADC supports an offset calibration function in which the OFSTRIM register is automatically updated. To perform offset calibration:

1. Configure for the desired averaging via CTRL[CAL_AVGS].
2. Initiate Offset Calibration by setting CTRL[CALOFS].
3. Poll the STAT[CAL_RDY] flag. When STAT[CAL_RDY] is asserted, the offset calibration function has completed, and the OFSTRIM register has updated.

41.5.1.2 ADC Calibration

ADC includes hardware calibration logic in which the A-side and B-side converters are calibrated for gain error and linearity error correction of the raw conversion result. GCR0[GCALR] and the CAL_GAR registers control calibration for the A-side converter. GCR1[GCALR] and the CAL_GBR registers control calibration for the B-side converter. ADC supports a calibration function in which the CAL_GAR and CAL_GBR registers are automatically updated. The calibration function also updates GCC0[GAIN_CAL] (associated with A-side calibration) and GCC1[GAIN_CAL] (associated with B-side calibration). For A-side calibration, a software calculation is required to derive GCR0[GCALR] from GCC0[GAIN_CAL]. For B-side calibration, a corresponding calculation is required to derive GCR1[GCALR] from GCC1[GAIN_CAL].

To complete calibration setup:

1. Execute Offset calibration steps. The Offset Trim Register (OFSTRIM) is used during calibration to trim for comparator offset voltage.
2. Configure the averaging via CTRL[CAL_AVGS].
3. Initiate the calibration routine by writing 1 to CTRL[CAL_REQ]. CTRL[CAL_REQ] remains 1 until the CAL routine has been accepted by the ADC. After acceptance, CTRL[CAL_REQ] automatically becomes 0.
4. The A-side calibration data is updated first. Poll the GCR0[RDY] flag. When it is asserted, the hardware controlled A-side calibration operation is complete, and CAL_GAR and GCC0[GAIN_CAL] registers are updated. The updated value in GCC0[GAIN_CAL] is required for further software processing described in the following steps.
5. Read GCC0[GAIN_CAL] and store for use in the gain_adjustment calculation.
6. Calculate the A-side gain_adjustment: (131072)/(131072-GCC0[GAIN_CAL]). GCC0[GAIN_CAL] is a 16-bit unsigned value. This calculation results in a floating point value between 1 and 2.
7. The integer value of the gain_adjustment calculation is always 1 and can be discarded. The fractional component must be stored to GCR0[GCALR]. Write the fractional component value to GCR0[GCALR].
8. Execute the same calculation for the B-side converter. Poll the GCC1[RDY] flag. When it is asserted, the hardware controlled B-side calibration operation is complete, and CAL_GBR and GCC1[GAIN_CAL] registers are updated. The updated value in GCC1[GAIN_CAL] is needed for further software processing.
9. Read the GCC1[GAIN_CAL] register and store for use in the gain_adjustment calculation.
10. Calculate the B-side gain_adjustment = (131072)/(131072-GCC1[GAIN_CAL]). Formatting for the value is the same as the A-side.
11. Like the A-side, round the fractional component of the B-side gain_adjustment to 16-bits and store it in GCR1[GCALR].

12. Once GCR0[GCALR] and GCR1[GCALR] contain the results from the gain_adjustment calculations, set the GCR0[RDY] and GCR1[RDY] flags to indicate they are valid. It is acceptable for the GCRn[GCALR] and corresponding GCRn[RDY] field to be updated on the same write cycle.

After completing the steps above, the calibration sequence is complete and the STAT[CAL_RDY] flag is set. The STAT[CAL_RDY] flag remains set until the user resets the system or requests a new calibration sequence.

When STAT[CAL_RDY] is set, ADC is configured to run in calibrated mode. Each conversion uses a combination of linearity and gain calibration results to correct SAR data. Calibration conversion latency is required to process each sample. However, due to the pipelined nature of data and control sequences, each conversion can still be initiated without experiencing this calibration delay.

41.5.1.3 Calibration General A-Side and B-Side Widths

The general calibration value registers CAL_GARn and CAL_GBRn have non-uniform widths. The following table defines the bit widths of each register.

Table 368. Calibration General Widths

Element CAL_GxR[N]	Width (Bits)
N = 00h, 20h	11
N = 01h	12
N = 02h–03h	13
N = 0x04–07h	14
N = 0x08–0Fh	15
N = 0x10–1Fh	16

These registers are typically updated automatically during the self-calibration sequence. To reduce the latency associated with ADC setup, the CAL_GxR values from a calibration sequence can be stored in non-volatile memory after an initial calibration. These values can then be written to the CAL_GxR registers via software prior to the first ADC conversion. If these registers are set to values not generated by the calibration function, the linearity error specifications may not be met.

NOTE

These values can only be written in a single access, byte accesses are not supported.

41.6 ADC register descriptions

This section describes the ADC registers.

41.6.1 ADC memory map

ADC0 base address: 4010_D000h

ADC1 base address: 4010_E000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	R	0200_2C0Bh
4h	Parameter Register (PARAM)	32	R	0F0F_1004h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
10h	Control Register (CTRL)	32	RW	0000_0000h
14h	Status Register (STAT)	32	RW	0000_0000h
18h	Interrupt Enable Register (IE)	32	RW	0000_0000h
1Ch	DMA Enable Register (DE)	32	RW	0000_0000h
20h	Configuration Register (CFG)	32	RW	0080_0000h
24h	Pause Register (PAUSE)	32	RW	0000_0000h
34h	Software Trigger Register (SWTRIG)	32	RW	0000_0000h
38h	Trigger Status Register (TSTAT)	32	RW	0000_0000h
40h	Offset Trim Register (OFSTRIM)	32	RW	0000_0000h
A0h - ACh	Trigger Control Register (TCTRL0 - TCTRL3)	32	RW	0000_0000h
E0h - E4h	FIFO Control Register (FCTRL0 - FCTRL1)	32	RW	0000_0000h
F0h - F4h	Gain Calibration Control (GCC0 - GCC1)	32	R	0000_0000h
F8h - FCh	Gain Calculation Result (GCR0 - GCR1)	32	RW	0000_0000h
100h	Command Low Buffer Register (CMDL1)	32	RW	0000_0000h
104h	Command High Buffer Register (CMDH1)	32	RW	0000_0000h
108h	Command Low Buffer Register (CMDL2)	32	RW	0000_0000h
10Ch	Command High Buffer Register (CMDH2)	32	RW	0000_0000h
110h	Command Low Buffer Register (CMDL3)	32	RW	0000_0000h
114h	Command High Buffer Register (CMDH3)	32	RW	0000_0000h
118h	Command Low Buffer Register (CMDL4)	32	RW	0000_0000h
11Ch	Command High Buffer Register (CMDH4)	32	RW	0000_0000h
120h	Command Low Buffer Register (CMDL5)	32	RW	0000_0000h
124h	Command High Buffer Register (CMDH5)	32	RW	0000_0000h
128h	Command Low Buffer Register (CMDL6)	32	RW	0000_0000h
12Ch	Command High Buffer Register (CMDH6)	32	RW	0000_0000h
130h	Command Low Buffer Register (CMDL7)	32	RW	0000_0000h
134h	Command High Buffer Register (CMDH7)	32	RW	0000_0000h
138h	Command Low Buffer Register (CMDL8)	32	RW	0000_0000h
13Ch	Command High Buffer Register (CMDH8)	32	RW	0000_0000h
140h	Command Low Buffer Register (CMDL9)	32	RW	0000_0000h
144h	Command High Buffer Register (CMDH9)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
148h	Command Low Buffer Register (CMDL10)	32	RW	0000_0000h
14Ch	Command High Buffer Register (CMDH10)	32	RW	0000_0000h
150h	Command Low Buffer Register (CMDL11)	32	RW	0000_0000h
154h	Command High Buffer Register (CMDH11)	32	RW	0000_0000h
158h	Command Low Buffer Register (CMDL12)	32	RW	0000_0000h
15Ch	Command High Buffer Register (CMDH12)	32	RW	0000_0000h
160h	Command Low Buffer Register (CMDL13)	32	RW	0000_0000h
164h	Command High Buffer Register (CMDH13)	32	RW	0000_0000h
168h	Command Low Buffer Register (CMDL14)	32	RW	0000_0000h
16Ch	Command High Buffer Register (CMDH14)	32	RW	0000_0000h
170h	Command Low Buffer Register (CMDL15)	32	RW	0000_0000h
174h	Command High Buffer Register (CMDH15)	32	RW	0000_0000h
200h - 238h	Compare Value Register (CV1 - CV15)	32	RW	0000_0000h
300h - 304h	Data Result FIFO Register (RESFIFO0 - RESFIFO1)	32	R	0000_0000h
400h	Calibration General A-Side Registers (CAL_GAR0)	32	RW	0000_0000h
404h	Calibration General A-Side Registers (CAL_GAR1)	32	RW	0000_0000h
408h - 40Ch	Calibration General A-Side Registers (CAL_GAR2 - CAL_GAR3)	32	RW	0000_0000h
410h - 41Ch	Calibration General A-Side Registers (CAL_GAR4 - CAL_GAR7)	32	RW	0000_0000h
420h - 43Ch	Calibration General A-Side Registers (CAL_GAR8 - CAL_GAR15)	32	RW	0000_0000h
440h - 47Ch	Calibration General A-Side Registers (CAL_GAR16 - CAL_GAR31)	32	RW	0000_0000h
480h	Calibration General A-Side Registers (CAL_GAR32)	32	RW	0000_0000h
500h	Calibration General B-Side Registers (CAL_GBR0)	32	RW	0000_0000h
504h	Calibration General B-Side Registers (CAL_GBR1)	32	RW	0000_0000h
508h - 50Ch	Calibration General B-Side Registers (CAL_GBR2 - CAL_GBR3)	32	RW	0000_0000h
510h - 51Ch	Calibration General B-Side Registers (CAL_GBR4 - CAL_GBR7)	32	RW	0000_0000h
520h - 53Ch	Calibration General B-Side Registers (CAL_GBR8 - CAL_GBR15)	32	RW	0000_0000h
540h - 57Ch	Calibration General B-Side Registers (CAL_GBR16 - CAL_GBR31)	32	RW	0000_0000h
580h	Calibration General B-Side Registers (CAL_GBR32)	32	RW	0000_0000h

41.6.2 Version ID Register (VERID)

Offset

Register	Offset
VERID	0h

Function

Indicates the version integrated for this instance on the device and the inclusion of optional features.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	NUM_FIFO			NUM_SEC	CALOFSI	IADCKI	VR1RNGI	0	CSW			MVI	0	DIFFEN	RES
W																
Reset	0	0	1	0	1	1	0	0	0	0	0	0	1	0	1	1

Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15 —	Reserved
14-12 NUM_FIFO	Number of FIFOs Indicates the number of result FIFOs implemented in the design. 000b - N/A 001b - One 010b - Two 011b - Three 100b - Four

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 NUM_SEC	Number of Single-Ended Outputs Supported Indicates the number of single-ended channels which can be processed simultaneously. 0b - One 1b - Two
10 CALOFSI	Calibration Function Implemented Indicates whether ADC contains hardware calibration functions. When supported, CTRL[CAL_REQ] can be used to request the calibration routine. 0b - Not implemented 1b - Implemented
9 IADCKI	Internal ADC Clock Implemented Indicates whether this implementation of the ADC block includes an internal clock source. 0b - Not implemented 1b - Implemented
8 VR1RNGI	Voltage Reference 1 Range Control Bit Implemented Indicates whether a control bit is implemented to select the input voltage range on Voltage Reference Option 1. 0b - Range control not required. 1b - Range control required.
7 —	Reserved
6-4 CSW	Channel Scale Width Indicates whether channel scaling is supported. When supported, each command buffer has a control field (CMDLn[CSCALE]) for setting input scaling. 000b - Not supported. 001b - Supported with one-bit CSCALE control field. 110b - Supported with six-bit CSCALE control field.
3 MVI	Multiple Vref Implemented Indicates whether multiple voltage reference high (VREFH) inputs are supported. When multiple voltage references are supported, CFG[REFSEL] selects voltage reference high options. 0b - Single VREFH input supported. 1b - Multiple VREFH inputs supported.
2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1 DIFFEN	Differential Supported Indicates whether differential operation is supported. When supported, each command buffer has control fields (CMDLn[CTYPE]) for configuring for differential operation and expanding the number of supported channels from 32 to 64. 0b - Not supported 1b - Supported. CMDLn[CTYPE] controls fields implemented.
0 RES	Resolution Indicates the maximum accuracy supported. 0b - Up to 13-bit differential or 12-bit single-ended resolution supported. 1b - Up to 16-bit differential or 16-bit single-ended resolution supported. CMDLn[MODE] available for selecting the resolution of conversions for the associated command.

41.6.3 Parameter Register (PARAM)

Offset

Register	Offset
PARAM	4h

Function

The Parameter register indicates the size of several variable integration options for this instance on the device.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMD_NUM								CV_NUM							
W																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIFOSIZE								TRIG_NUM							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31-24 CMD_NUM	Command Buffer Number Indicates number of command buffers implemented.
23-16 CV_NUM	Compare Value Number Indicates number of compare value registers implemented.
15-8 FIFOSIZE	Result FIFO Depth Indicates the maximum number of conversion data words that can be stored in the result FIFO before an overflow occurs. 0000_0001b - 2 0000_0100b - 4 0000_1000b - 8 0001_0000b - 16 0010_0000b - 32 0100_0000b - 64
7-0 TRIG_NUM	Trigger Number Number of Triggers implemented.

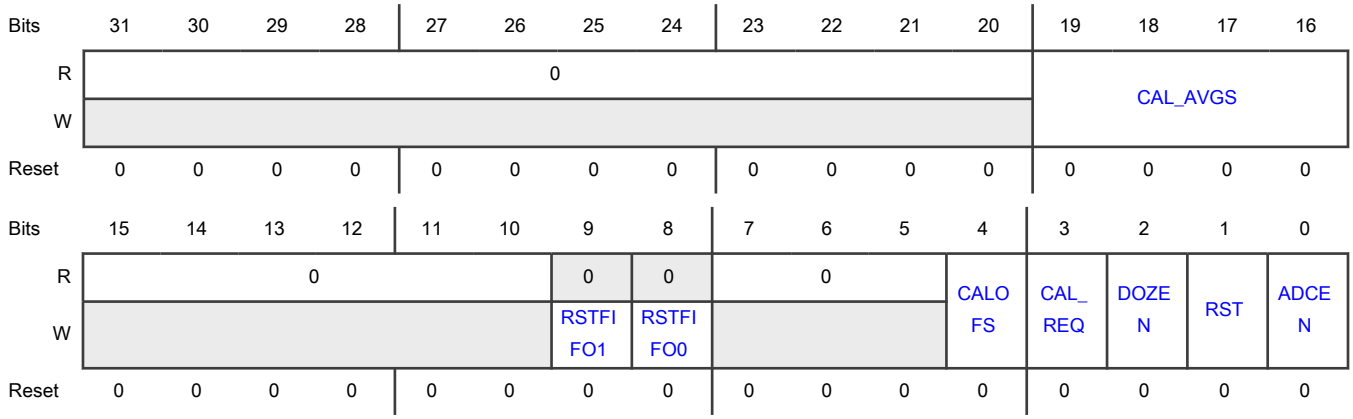
41.6.4 Control Register (CTRL)

Offset

Register	Offset
CTRL	10h

Function
Includes primary control bits.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 CAL_AVGS	Auto-Calibration Averages Selects how many ADC conversions are averaged to calculate each calibration value. Selecting a higher number of averages will lead to more accurate conversions after completing calibration. The CAL_AVGS bit field applies to both CALOFS and CAL_REQ calibration types. This value should be fixed when requesting and running calibration with CTRL[CAL_REQ] or CTRL[CALOFS]. 0000b - Single conversion. 0001b - 2 conversions averaged. 0010b - 4 conversions averaged. 0011b - 8 conversions averaged. 0100b - 16 conversions averaged. 0101b - 32 conversions averaged. 0110b - 64 conversions averaged. 0111b - 128 conversions averaged. 1000b - 256 conversions averaged. 1001b - 512 conversions averaged. 1010b - 1024 conversions averaged.
15-10 —	Reserved
9 RSTFIFO1	Reset FIFO 1 0b - No effect. 1b - FIFO 1 is reset.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 RSTFIFO0	Reset FIFO 0 0b - No effect. 1b - FIFO 0 is reset.
7-5 —	Reserved
4 CALOFS	Offset Calibration Request Indicates whether a request for a hardware calibration calculation has been made. Writing 1 to this field initiates a hardware offset calibration calculation. Any conversions in progress are completed before launching the offset calibration function. After being accepted, ADC calculates the OFSTRIM[OFSTRIM] value and updates the OFSTRIM register automatically. This field becomes 0 upon completion of the offset calibration calculation. STAT[CAL_RDY] indicates when the offset calibration routine has completed. 0b - Calibration function disabled 1b - Request for offset calibration function
3 CAL_REQ	Auto-Calibration Request Indicates whether a request for the hardware calibration routine has been made. Automatically becomes 0 when the system accepts the hardware calibration request. STAT[CAL_RDY] indicates when this calibration routine has been completed. 0b - No request made. 1b - Request has been made.
2 DOZEN	Doze Enable Controls system transition to low-power modes while ADC is converting. When 0, immediate entries to low-power modes are allowed and ADC conversion functions remain enabled. Any conversion in progress is not disrupted. The selected clock source provided from the on-chip clock source must be able to continue operating. When 1, the ADC waits for the current averaging iteration or FIFO storage to complete before acknowledging the low-power entry request. After entering the low-power state, ADC is kept inactive until the system exits the low-power state. When the system enters a low-power mode in which ADC operation is not supported, the DOZEN bit is ignored and ADC waits for the current transfer to complete any pending operation. 0b - ADC is enabled in low-power mode. 1b - ADC is disabled in low-power mode.
1 RST	Software Reset Resets all internal logic and registers, except the CTRL register. Remains 1 until cleared by software. 0b - ADC logic is not reset. 1b - ADC logic is reset.
0	ADC Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
ADCEN	0b - Disabled 1b - Enabled

41.6.5 Status Register (STAT)

Offset

Register	Offset
STAT	14h

Function

Provides the status of the ADC module.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				CMDACT				0						TRGACT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				ADC_ACT...	CAL_RDY	TCOMP_I...	TEXC_INT	0				FOF1	RDY1	FOF0	RDY0
W							W1C	W1C					W1C		W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 CMDACT	Command Active Indicates the command actively being processed. Commands are only shown here when they are actively being processed by the SAR routine. Use with STAT[ADC_ACTIVE] to determine which conversion is running. 0000b - No command currently in progress. 0001b - Command 1 currently being executed.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0010b - Command 2 currently being executed. 0011b-1111b - Associated command number currently being executed.
23-18 —	Reserved
17-16 TRGACT	Trigger Active Indicates the trigger actively being processed. This field can be used to determine which trigger is running through a trigger delay or being converted by the SAR routine. The command associated with this field is running through a conversion when STAT[CMDACT] is not zero. 00b - Command (sequence) associated with Trigger 0 currently being executed. 01b - Command (sequence) associated with Trigger 1 currently being executed. 10b - Command (sequence) associated with Trigger 2 currently being executed. 11b - Command (sequence) associated with Trigger 3 currently being executed.
15-12 —	Reserved
11 ADC_ACTIVE	ADC Active Indicates whether the module is processing a conversion, or has pending triggers to service. 0b - ADC is idle. There are no pending triggers to service and no active commands are being processed. 1b - ADC is processing a conversion, running through the power-up delay, or servicing a trigger.
10 CAL_RDY	Calibration Ready Indicates whether the ADC request for calibration (CTRL[CAL_REQ] or CTRL[CALOFS]) has been completed and the results are ready. This flag is automatically cleared when a new hardware calibration or OFSTRIM request is made. 0b - Calibration is incomplete or has not been run. 1b - ADC is calibrated.
9 TCOMP_INT	Interrupt Flag For Trigger Completion Indicates when a trigger sequence has been completed (all associated commands have been run). IE[TCOMP_IE] must be 1 for the specific trigger source to flag an interrupt upon completion. 0b - Either IE[TCOMP_IE] = 0, or no trigger sequences have run to completion. 1b - Trigger sequence has been completed and all data is stored in the associated FIFO.
8 TEXC_INT	Interrupt Flag For High-Priority Trigger Exception Indicates when a high-priority trigger exception has occurred and CFG[TRES] = 0. This flag only asserts if trigger exception interrupts are enabled (IE[TEXC_IE] = 1h). This flag can be used with TSTAT[TEXC_NUM] to resolve which trigger source was interrupted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No trigger exceptions have occurred. 1b - A trigger exception has occurred and is pending acknowledgment.
7-4 —	Reserved
3 FOF1	Result FIFO1 Overflow Flag Indicates that more data has been written to the Result FIFO 1 than it can hold. The newer data is not stored and the FIFO holds the original contents. This flag asserts regardless of the value of IE[FOFIE1]. However, an interrupt request is issued only if IE[FOFIE1] is 1. 0b - No result FIFO1 overflow has occurred since the last time that the flag was cleared. 1b - At least one result FIFO1 overflow has occurred since the last time that the flag was cleared.
2 RDY1	Result FIFO1 Ready Flag Indicates when the number of valid data words in the result FIFO 1 is greater than the level set in FCTRL0[FWMARK]. This flag asserts regardless of the value of IE[FWMIE1]. However, an interrupt request or DMA request occurs only when the associated control bit (IE[FWMIE1] and DE[FWMDE1]) is set. This flag is cleared when FCTRL0[FCOUNT] (which decrements on each read of the RESFIFO register) is less than or equal to the level set in FCTRL0[FWMARK]. 0b - Not above watermark 1b - Above watermark
1 FOF0	Result FIFO 0 Overflow Flag Indicates that more data has been written to the Result FIFO 0 than it can hold. The newer data is not stored and the FIFO holds the original contents. This flag asserts regardless of the value of IE[FOFIE0]. However, an interrupt request is issued only if IE[FOFIE0] is 1. 0b - No result FIFO 0 overflow has occurred since the last time that the flag was cleared. 1b - At least one result FIFO 0 overflow has occurred since the last time that the flag was cleared.
0 RDY0	Result FIFO 0 Ready Flag Indicates when the number of valid data words in the result FIFO 0 is greater than the watermark set in FCTRL0[FWMARK]. This flag asserts regardless of the value of IE[FWMIE0]. However, an interrupt request or DMA request occurs only when the associated control field (IE[FWMIE0] and DE[FWMDE0]) is 1. This flag is cleared when the FCTRL0[FCOUNT] (which decrements on each read of the RESFIFO register) is less than or equal to the level set in FCTRL0[FWMARK]. 0b - Not above watermark 1b - Above watermark

41.6.6 Interrupt Enable Register (IE)

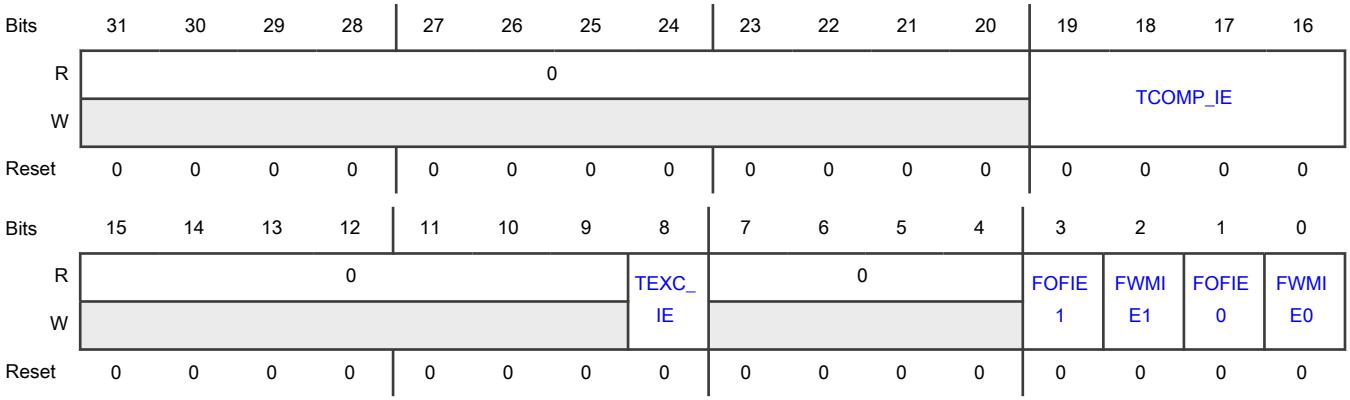
Offset

Register	Offset
IE	18h

Function

Includes system interrupt masking control bits.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 TCOMP_IE	Trigger Completion Interrupt Enable Enables generation of interrupt requests to indicate when complete trigger command sequences are executed. Each bit in TCOMP_IE corresponds to a trigger source. (TCOMP_IE[0] corresponds to trigger 0, and so on.) All results are stored in the FIFO when a sequence is considered complete. 0000b - All disabled 0001b - Trigger completion interrupts are enabled for trigger source 0 only. 0010b - Trigger completion interrupts are enabled for trigger source 1 only. 0011b-1110b - Associated trigger completion interrupts are enabled. 1111b - All enabled
15-9 —	Reserved
8	Trigger Exception Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TEXC_IE	Enables ADC to assert an interrupt request when a high-priority trigger exception occurs. TSTAT[TEXC_NUM] contains the value of the corresponding trigger affected by the exception. 0b - Disabled 1b - Enabled
7-4 —	Reserved
3 FOFIE1	Result FIFO1 Overflow Interrupt Enable Enables generation of overflow interrupt requests when FOF1 flag is asserted. 0b - Disabled 1b - Enabled
2 FWMIE1	FIFO1 Watermark Interrupt Enable Enables generation of watermark interrupt requests when STAT[RDY1] flag is asserted. 0b - Disabled 1b - Enabled
1 FOFIE0	Result FIFO 0 Overflow Interrupt Enable Enables generation of overflow interrupt requests when FOF flag is asserted. 0b - Disabled 1b - Enabled
0 FWMIE0	FIFO 0 Watermark Interrupt Enable Enables generation of watermark interrupt requests when STAT[RDY0] flag is asserted. 0b - Disabled 1b - Enabled

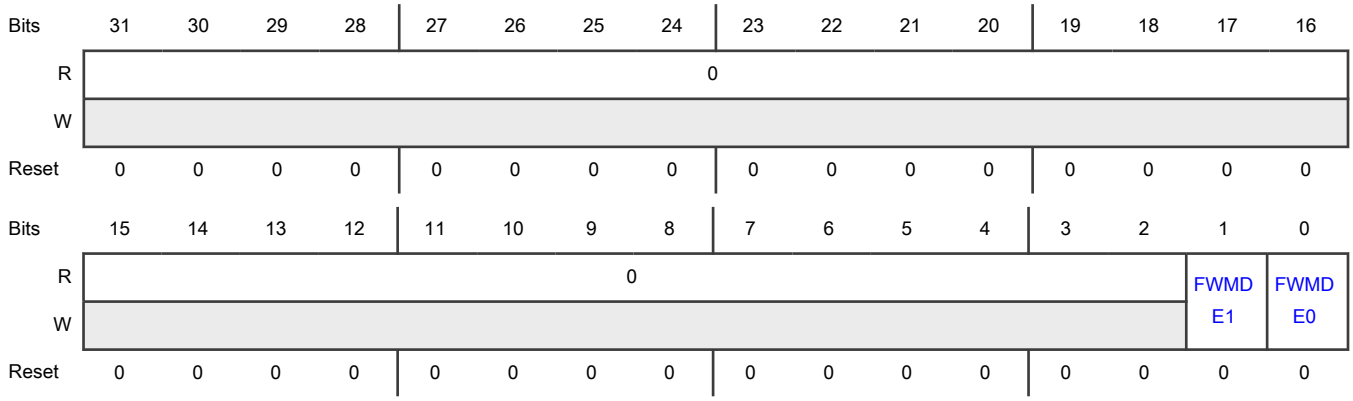
41.6.7 DMA Enable Register (DE)

Offset

Register	Offset
DE	1Ch

Function
Includes DMA request-masking control bits.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 FWMDE1	FIFO1 Watermark DMA Enable Enables generation of DMA requests when STAT[RDY1] flag is asserted. 0b - Disabled 1b - Enabled
0 FWMDE0	FIFO 0 Watermark DMA Enable Enables generation of DMA requests when STAT[RDY0] flag is asserted. 0b - Disabled 1b - Enabled

41.6.8 Configuration Register (CFG)

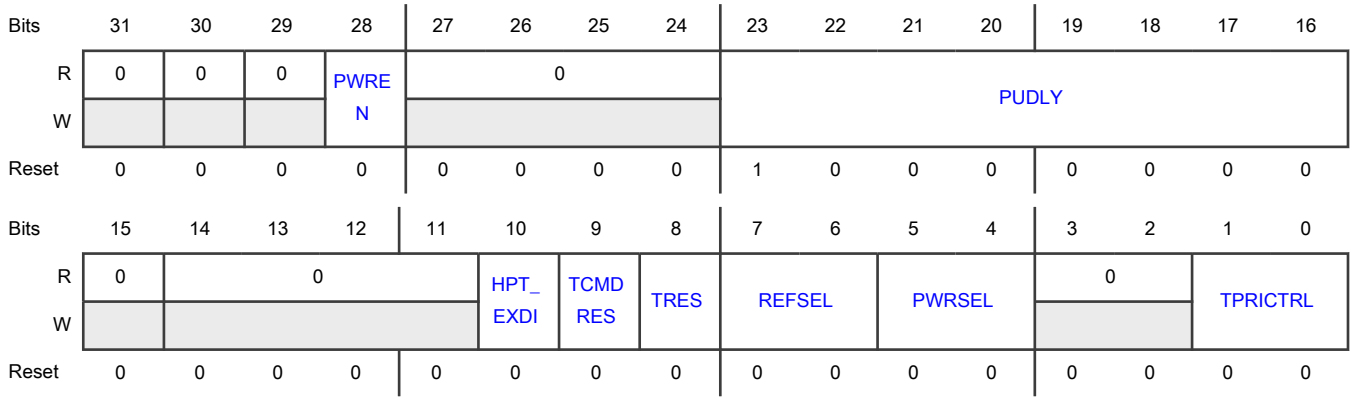
Offset

Register	Offset
CFG	20h

Function

Controls ADC functions common to all commands. When CTRL[ADCEN] = 1, this register cannot be changed and writes to this register are ignored.

Diagram



Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28 PWREN	ADC Analog Pre-Enable Enables the ADC analog circuits. When setting this field, user code should delay for a period exceeding the analog startup time of $t_{ADCSTUP}$ before enabling ADC for operation. The module is still operational even when this field is 0, but command execution start is delayed for a period defined by CFG[PUDLY]. See the device data sheet for ADC idle and active power consumption parameters. 0b - ADC analog circuits are only enabled while conversions are active. Analog startup delays affect performance. 1b - ADC analog circuits are pre-enabled and ready to execute conversions without startup delays, at the cost of higher DC current consumption. A single power-up delay (CFG[PUDLY]) is executed immediately once PWREN is set. No detected triggers begin ADC operation until the power-up delay time has passed. After this initial delay expires, the analog circuits remain pre-enabled, and no additional delays are executed.
27-24 —	Reserved
23-16 PUDLY	Power-up Delay Defines the power-up delay executed after an initial trigger wakes ADC from its idle state. Must be programmed to a non-zero value to enable the ADC. Depending on the value of CFG[PWREN], the delay is executed in one of two modes:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• When CFG[PWREN] = 0, the ADC analog circuits are only powered on while the module is active. The power-up delay allows time for the analog circuits to stabilize after being powered on. The startup delay count of (PUDLY * 4) ADCK cycles must result in a longer delay than the analog startup time of $t_{ADCSTUP}$. Accuracy of the initial conversions after activation is degraded if CFG[PUDLY] is set to too small a value. After active conversions, if no subsequent conversions are pending, the ADC analog circuits are automatically reverted to their low-power idle state. When ADC is awakened with a later trigger, the power-up delay runs again. • When CFG[PWREN] = 1 prior to ADC activation, the analog circuits are pre-enabled and the activation delay defined by CFG[PUDLY] is executed immediately. After the delay has been executed, the analog circuits remain enabled regardless of ADC activity. This configuration begins conversions immediately after trigger event detection, at the cost of increased DC power consumption in the analog circuits. **NOTE** ADC does not begin an initial conversion until the power-up delay has executed, regardless of the value of CFG[PWREN].
15 —	Reserved
14-11 —	Reserved
10 HPT_EXDI	High-Priority Trigger Exception Disable Disables high-priority trigger exceptions. See Trigger detect and command execution for a detailed description on trigger exception handling. When 1, exceptions are disabled and CFG[TCMDRES], CFG[TRES], and CFG[TPRICTRL] are ignored. 0b - Enabled 1b - Disabled
9 TCMDRES	Trigger Command Resume Determines where a trigger sequence resumes when interrupted by a high-priority trigger exception. To use this field, CFG[TRES] must be 1. 0b - Trigger sequence automatically restarted. 1b - Trigger sequence resumed from the command that was executed prior to the exception.
8 TRES	Trigger Resume Enable Determines whether trigger sequences interrupted by a high-priority exception are automatically resumed or restarted. If 1, the sequence can be resumed along command boundaries or restarted from the beginning of a sequence, as determined by CFG[TCMDRES]. If 0, interrupted triggers can be resumed via software by monitoring STAT[TEXC_INT] (or via ISR handling for trigger exception interrupt when IE[TEXC_IE] = 1). Software determines which trigger was interrupted

Table continues on the next page...

Table continued from the previous page...

Field	Function
	by reading TSTAT[TEXC_NUM] and restarts the trigger by writing 1 to the corresponding field in the SWTRIG register. 0b - Not automatically resumed or restarted 1b - Automatically resumed or restarted
7-6 REFSEL	Voltage Reference Selection Selects the voltage reference high used for conversions. **NOTE** See the chip-specific ADC information for voltage reference options specific to this packaged device. 00b - Option 1 01b - Option 2 10b - Option 3 11b - Reserved
5-4 PWRSEL	Power Configuration Select Configures the module for power and performance. In the high-power setting, the highest conversion rates are possible. See the device data sheet for power and performance capabilities for each setting. 0xb - Low power 1xb - High power
3-2 —	Reserved
1-0 TPRICTRL	ADC Trigger Priority Control Controls how higher-priority trigger exceptions are handled when they are received during command processing. See Trigger detect and command execution for a detailed explanation of trigger event handling. 00b - Current conversion is aborted and the new command specified by the trigger is started. 01b - Current command is stopped after completing the current conversion. If averaging is enabled, the averaging loop is completed. CMDHn[LOOP] is ignored and the higher-priority trigger is serviced. 10b - Current command is completed (averaging, looping, compare) before servicing the higher-priority trigger. 11b - Reserved

41.6.9 Pause Register (PAUSE)

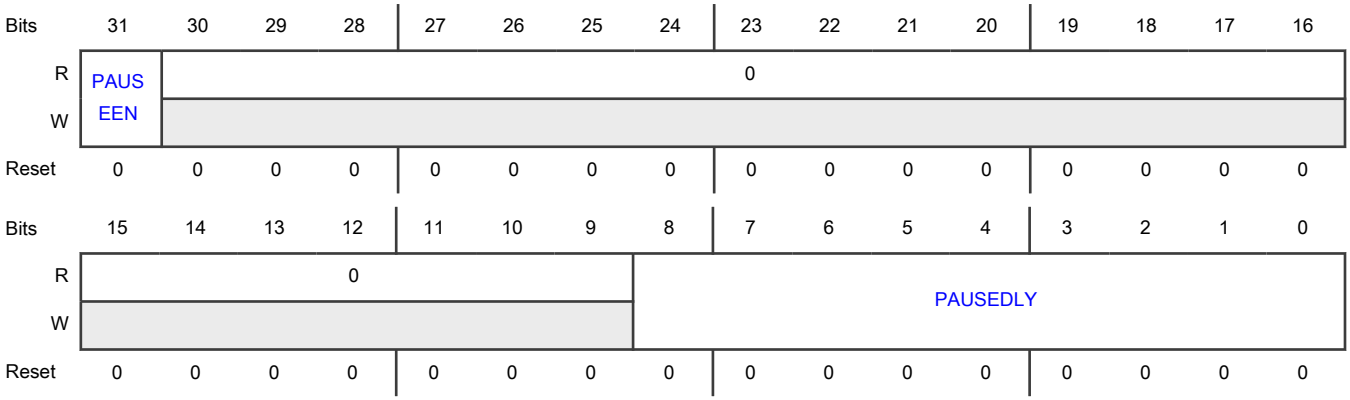
Offset

Register	Offset
PAUSE	24h

Function

Controls the optional inserted delay between conversions. When CTRL[ADCEN] = 1, this register should not be modified.

Diagram



Fields

Field	Function
31 PAUSEEN	Pause Enable Enables the ADC pausing function. When enabled, a programmable delay is inserted during command execution sequencing: • Between LOOP iterations. • Between commands in a sequence. • Between conversions, when a command is executing in compare-until-true configuration. **NOTE** CMDHn[WAIT_TRIG] and PAUSE are mutually exclusive. CMDHn[WAIT_TRIG] takes priority over PAUSE between commands when both are enabled. 0b - Disabled 1b - Enabled
30-9 —	Reserved
8-0 PAUSEDLY	Pause Delay Controls the duration of pauses during command execution sequencing when PAUSE[PAUSEEN] = 1. The pause delay is a count of (PAUSEDLY * 4) ADCK cycles.

41.6.10 Software Trigger Register (SWTRIG)

Offset

Register	Offset
SWTRIG	34h

Function

Initiates software-triggered conversions when written to. Writes to this register are ignored while CTRL[ADCEN] = 0. There is an approximately three ADC-clock-cycle synchronization delay between asserting CTRL[ADCEN] and when SWTRIG can be accepted.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												0	0	0	0
W													SWT3	SWT2	SWT1	SWT0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-4 —	Reserved
3 SWT3	Software Trigger 3 Write 1 to SWT3 generates a trigger 3 event. Writing 1 to SWT3 is ignored while the trigger 3 event is being serviced or is pending. 0b - No trigger 3 event generated. 1b - Trigger 3 event generated.
2 SWT2	Software Trigger 2 Write 1 to SWT2 generates a trigger 2 event. Writing 1 to SWT2 is ignored while the trigger 2 event is being serviced or is pending. 0b - No trigger 2 event generated. 1b - Trigger 2 event generated.
1	Software Trigger 1

Table continues on the next page...

Table continued from the previous page...

Field	Function
SWT1	Write 1 to SWT1 generates a trigger 1 event. Writing 1 to SWT1 is ignored while the trigger 1 event is being serviced or is pending. 0b - No trigger 1 event generated. 1b - Trigger 1 event generated.
0 SWT0	Software Trigger 0 Generates a trigger 0 event. Writing 1 to this field is ignored while the trigger 0 event is being serviced or is pending. 0b - No trigger 0 event generated. 1b - Trigger 0 event generated.

41.6.11 Trigger Status Register (TSTAT)

Offset

Register	Offset
TSTAT	38h

Function

Contains status flags to indicate when trigger sequences have been completed or interrupted by a high-priority trigger exception. Each field in this register is set by hardware and cleared by software.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												TCOMP_FLAG			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												TEXC_NUM			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
19-16 TCOMP_FLAG	Trigger Completion Flag Indicates which triggers have been completed. Each bit in this field corresponds to a trigger. When a trigger sequence has completed, the corresponding bit in TCOMP_FLAG is set. For example, if trigger 3 has been completed, then bit 19 is set in the TSTAT register. This register is active only if the corresponding bit in IE[TCOMP_IE] = 1. **NOTE** A synchronization delay may be added to the end of the final conversion in a sequence, prior to this flag being set. This delay can range from two to four ADC CLK cycles. 0000b - No triggers have been completed. Trigger completion interrupts are disabled. 0001b - Trigger 0 has been completed and trigger 0 has enabled completion interrupts. 0010b - Trigger 1 has been completed and trigger 1 has enabled completion interrupts. 0011b-1110b - Associated trigger sequence has completed and has enabled completion interrupts. 1111b - Every trigger sequence has been completed and every trigger has enabled completion interrupts.
15-4 —	Reserved
3-0 TEXC_NUM	Trigger Exception Number Indicates which triggers have been interrupted by exceptions. Each bit in this field corresponds to a trigger. When the corresponding trigger sequence is interrupted by a high-priority trigger exception, the corresponding bit is set. For example, if trigger 3 has been interrupted, then bit 3 is set in this register. This register is active regardless of the value in IE[TEXC_IE]. 0000b - No triggers have been interrupted by a high-priority exception. Or CFG[TRES] = 1. 0001b - Trigger 0 has been interrupted by a high-priority exception. 0010b - Trigger 1 has been interrupted by a high-priority exception. 0011b-1110b - Associated trigger sequence has interrupted by a high-priority exception. 1111b - Every trigger sequence has been interrupted by a high-priority exception.

41.6.12 Offset Trim Register (OFSTRIM)

Offset

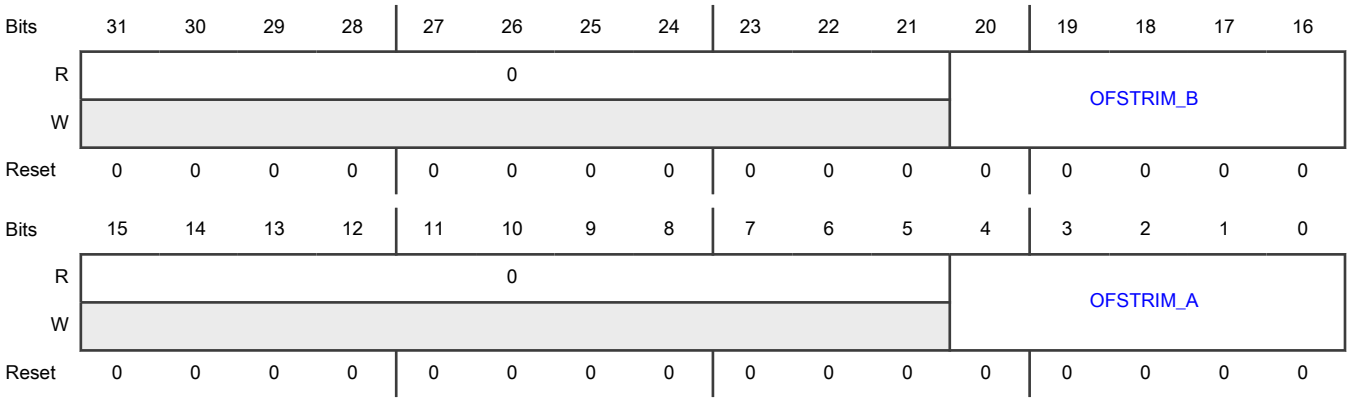
Register	Offset
OFSTRIM	40h

Function

Used to trim for offset. ADC supports a calibration step in which the ADC determines the value needed in the OFSTRIM register.

To determine the value to put in the OFSTRIM register, write 1 to CTRL[CALOFS]. This setting automatically begins a sequence to calculate this value. Once the sequence has completed, the OFSTRIM register is updated with a signed value between -16 and 15. This value is used to minimize offset during normal operation.

Diagram



Fields

Field	Function
31-21 —	Reserved
20-16 OFSTRIM_B	Trim for Offset OFSTRIM is a 5-bit signed value between -16 and 15. This value applies to the ADC B-side conversion results.
15-5 —	Reserved
4-0 OFSTRIM_A	Trim for Offset OFSTRIM_A is a 5-bit signed value between -16 and 15. This value applies to the ADC A-side conversion results.

41.6.13 Trigger Control Register (TCTRL0 - TCTRL3)

Offset

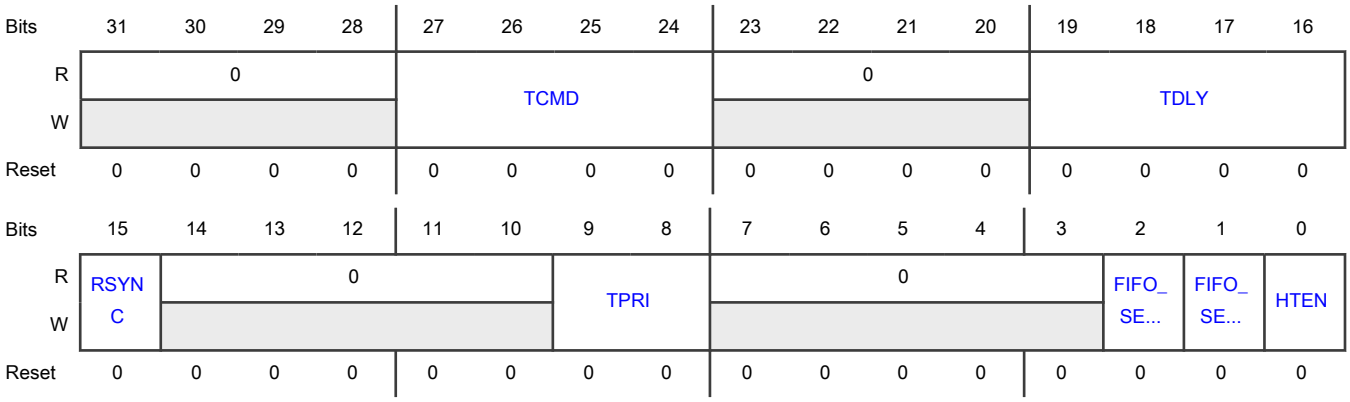
Register	Offset
TCTRL0	A0h
TCTRL1	A4h
TCTRL2	A8h
TCTRL3	ACh

Function

Implements control fields associated with each trigger source. When ADC is executing commands, only one TCTRLn register controls ADC conversions.

Do not update the controlling TCTRLn register while ADC is active. Writing to a TCTRLn register while that trigger control register controls the ADC operation may cause unpredictable behavior.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 TCMD	Trigger Command Select Selects the command from the command buffer to execute upon detection of the associated trigger event. When a higher-priority trigger is received while ADC is converting, the command in this register associated with the new trigger is executed. This execution is done under the control of CFG[TPRICTRL]. See Trigger detect and command execution for details about the relationship between CFG[TPRICTRL] and TCMD. 0000b - Not a valid selection from the command buffer. Trigger event is ignored. 0001b - CMD1 0010b-1110b - Corresponding CMD is executed 1111b - CMD15
23-20 —	Reserved
19-16 TDLY	Trigger Delay Select Selects the trigger delay duration for the start of servicing a trigger event. Each trigger source has an associated programmable delay prior to beginning an initial conversion. When this field is 0, no delay is incurred. When this field is set to a non-zero value, the duration of the delay is 2^{TDLY} ADCK cycles.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 RSYNC	Trigger Resync Enables trigger resync. If 1, this trigger source is used as a resync trigger. A resync trigger can be configured to restart or abort a running trigger sequence (the target). Each resync trigger is configured to have a resync target. When a resync trigger is asserted: • The target sequence is aborted. • The target FIFO destinations are cleared. • The target sequence can be restarted depending on CFG[TRES]. Multiple conditions must be met for a resync trigger to execute properly: • The resync trigger must have higher priority than the resync target. • The resync trigger TCTRLn[RSYNC] must be set to 1b. • The resync target, specified by TCTRLn[TCMD], must be executing when the resync trigger is asserted. See Resync functionality for more information. 0b - Disable 1b - Enable
14-10 —	Reserved
9-8 TPRI	Trigger Priority Setting Sets the priority of the associated trigger source. If two or more triggers have the same priority level, the lower-order trigger event has the higher priority. If Trigger 0 and Trigger 1 are pending triggers and TCTRL0[TPRI] is configured the same as TCTRL1[TPRI], the Trigger 0 command is serviced first. 00b - Highest priority, Level 1 01b-10b - Set to corresponding priority level. 11b - Lowest priority, Level 4
7-3 —	Reserved
2 FIFO_SEL_B	SAR Result Destination for Channel B Indicates the FIFO to which SAR results are written for Channel B. This field is only used in dual single-ended mode (CMDLn[CTYPE] = 11b). In this mode, the SAR result from Channel B is written to the FIFO number specified by this field. See Sampling Modes for more information. 0b - FIFO 0 1b - FIFO 1
1 FIFO_SEL_A	SAR Result Destination for Channel A

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates the FIFO to which SAR results are written for Channel A. This field provides the destination for all single-ended and differential conversions and for all A-side conversions in dual single-ended mode. Conversion results are stored in the FIFO number specified, under the following conditions: • Single-ended mode: CMDLn[CTYPE] = 00b or CMDLn[CTYPE] = 01b. • Differential mode: CMDLn[CTYPE] = 10b. • Dual single-ended mode: CMDLn[CTYPE] = 11b. 0b - FIFO 0 1b - FIFO 1
0 HTEN	Trigger Enable Enables hardware trigger source to initiate conversion on the rising edge of the input trigger source. **NOTE** Enabling the hardware trigger does not disable software triggers. 0b - Disabled 1b - Enabled

41.6.14 FIFO Control Register (FCTRL0 - FCTRL1)

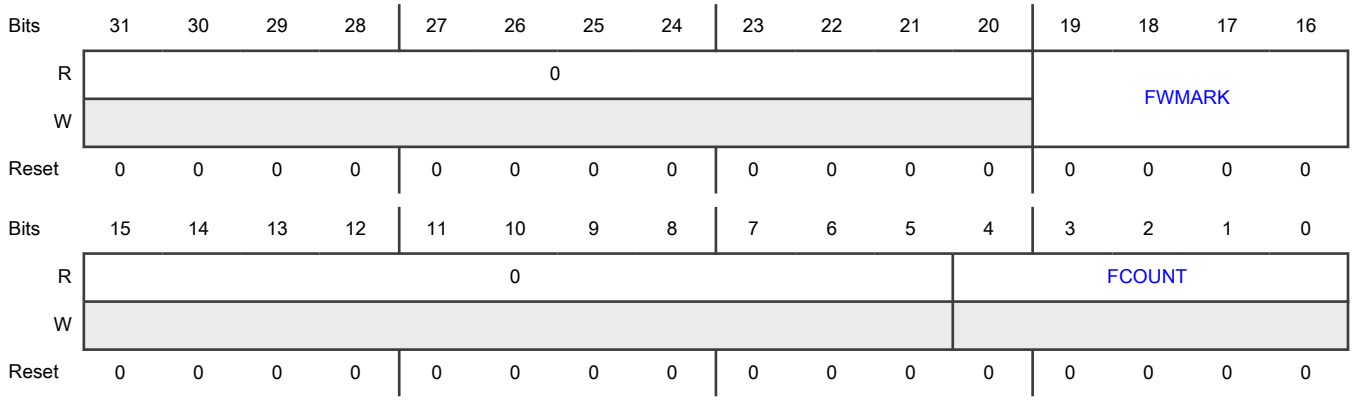
Offset

Register	Offset
FCTRL0	E0h
FCTRL1	E4h

Function

Contains control and status fields for each FIFO in the design. A programmable watermark can be set for each FIFO, which can be used to trigger an interrupt. In addition, the number of entries stored in each FIFO can be monitored by reading FCTRLn[FCOUNT].

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 FWMARK	Watermark Level Selection Selects the storage threshold for the ADC Result FIFO. When the number of data words stored in the FIFO is greater than this value, the STAT[RDY0] flag is asserted. When IE[FWMIEn] = 1, an interrupt request is generated. When DE[FWMDEn] = 1, a DMA request is generated.
15-5 —	Reserved
4-0 FCOUNT	Result FIFO Counter Indicates the number of data words stored in the result FIFO. This value may be used with PARAM[FIFOSIZE] to calculate how much room is left in the result FIFO. This field is incremented with each storage of new data into the result FIFO and decrements with each read of the result FIFO. The FIFO is reset by writing to CTRL[RSTFIFOn], which initializes FCTRLn[FCOUNT] to 0h.

41.6.15 Gain Calibration Control (GCC0 - GCC1)

Offset

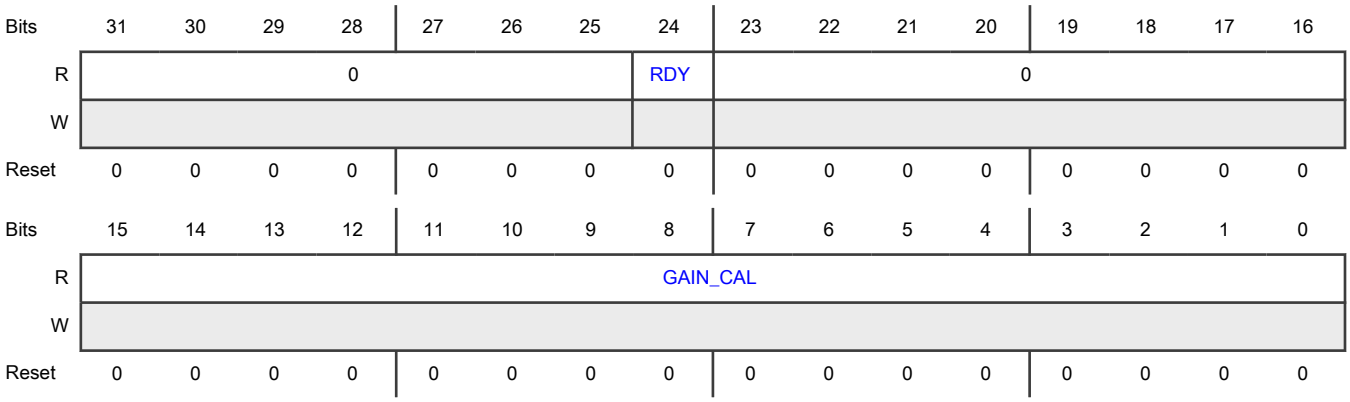
Register	Offset
GCC0	F0h
GCC1	F4h

Function

Holds intermediate values used as part of the calibration steps. GCC0 is associated with the A-side converter and GCC1 is associated with the B-side converter. GCCn[GAIN_CAL] is calculated with hardware and automatically updated during the calibration steps. Once the hardware calibration sequence has updated, GCCn[RDY] is asserted automatically.

Requesting a calibration routine automatically clears the GCCn[RDY] flag until the new GCCn[GAIN_CAL] value is calculated. To complete calibration, further software processing is needed, where GCCn[GAIN_CAL] is used to calculate the gain adjustment. The result is stored to GCRn[GCALR]. See Calibration functions.

Diagram



Fields

Field	Function
31-25 —	Reserved
24 RDY	Gain Calibration Value Valid Indicates whether the data stored in GCCn[GAIN_CAL] is valid and should be used to derive GCRn[GCALR]. If the data in GCCn[GAIN_CAL] is invalid, you can run the hardware calibration routine to correct this issue. 0b - Invalid 1b - Valid
23-16 —	Reserved
15-0 GAIN_CAL	Gain Calibration Value Indicates the calculated value of the gain calibration. As part of the hardware calibration steps, this field is automatically updated with a 16-bit unsigned number. It is used in the gain adjustment calculations to derive the value written to GCRn[GCALR]. See Calibration functions.

41.6.16 Gain Calculation Result (GCR0 - GCR1)

Offset

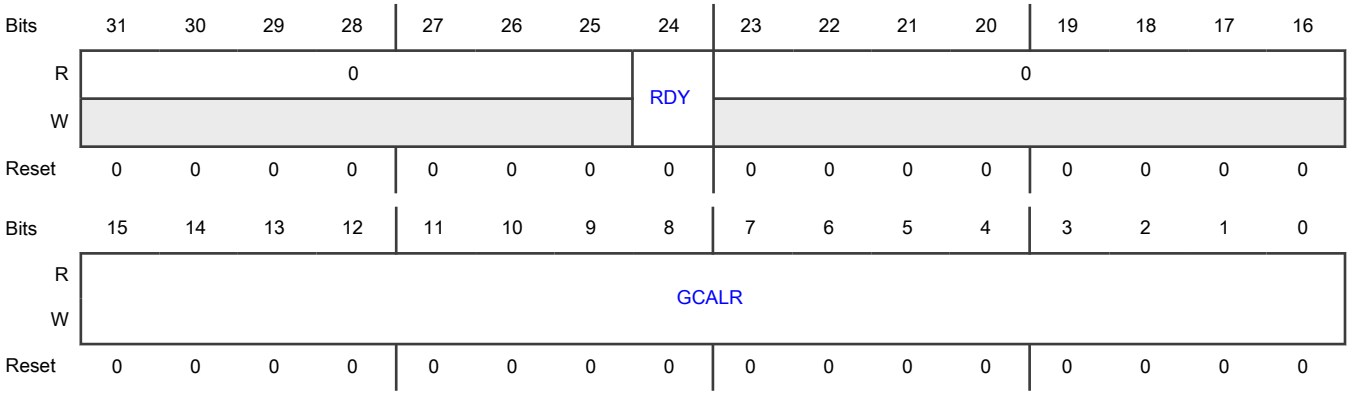
Register	Offset
GCR0	F8h
GCR1	FCh

Function

Used during ADC conversions for automated gain error adjustment. GCR0 is associated with the A-side converter and GCR1 is associated with the B-side converter. Determining the values to store to GCR0[GCALR] and GCR1[GCALR] is the result of software calculations described in setup steps in Calibration functions. Upon writing the GCRn[GCALR] value, the user should also write 1 to GCRn[RDY] to indicate that the gain adjustment result is valid.

A calibration sequence begins by writing 1 to CTRL[CAL_REQ], and the calibration sequence is not complete until GCRn[GCALR] is calculated and GCRn[RDY] is 1. The gain adjustment calculation produces a floating-point value between 1 and 2. GCRn[GCALR] holds the 16-bit fractional component of the gain adjustment calculation. To convert the GCRn[GCALR] value to the gain adjustment value in decimal format, use this formula: 1 + 0.5 * GCRn[15] + 0.25 * GCRn[14] + 0.125 * GCRn[13] + (and so on)

Diagram



Fields

Field	Function
31-25 —	Reserved
24 RDY	Gain Calculation Ready Indicates whether the data stored in GCRn[GCALR] is valid and is used for gain adjustment. GCRn[GCALR] must be written from memory or calculated each time the calibration routine is run. The user must write 1 to this GCRn[RDY] after writing valid data to GCRn[GCALR]. This field becomes 0 automatically when requesting a new calibration sequence. 0b - Invalid 1b - Valid
23-16 —	Reserved
15-0 GCALR	Gain Calculation Result Holds the 16-bit fractional component generated from the gain adjustment calculation during the auto-calibration routine.

41.6.17 Command Low Buffer Register (CMDL1 - CMDL15)

Offset

For a = 1 to 15:

Register	Offset
CMDLa	F8h + (a × 8h)

Function

Controls channel selection and conversion options. There are 15 command buffers (CMDn), each constructed from two 32-bit registers (CMDHn:CMDLn) that can be configured for different channel selection and varying conversion options. Any command buffer is selected and used as the controlling command by association with a trigger event via configuration of TCTRLn[TCMD]. When ADC is executing commands, only one of the CMD buffers controls ADC conversions. Do not update the controlling CMD buffer while ADC is active. A write to a CMD buffer while that CMD buffer controls the ADC operation may cause unpredictable behavior.

NOTE

The 15 command buffers are numbered [CMDH1:CMDL1] through [CMDH15:CMDL15]. In NXP-supplied header files, these buffers are likely to be defined as two 15-element arrays that are indexed from 0. For example, the type declaration would be:

```
unsigned int CMDH[15];
unsigned int CMDL[15];
```

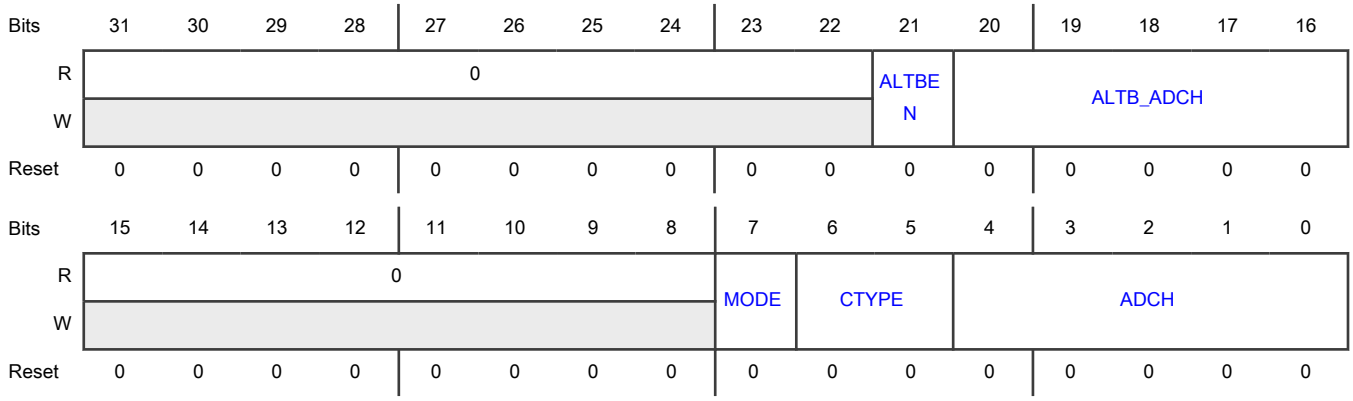
and software would access ADC0 CMDH1 as `ADC0->CMDH[0]` and ADC0 CMDL15 as `ADC0->CMDL[14]`.

The CMDLn[ADCH] and CMDLn[CTYPE] fields control selection of paired or individual input channels. Each ADC command independently makes a channel and conversion type selection. Each ADCH channel selection has an associated A-side and B-side input. Each ADCH pair can be converted in differential mode, but only limited pairs should be converted as differential channels (for example, adjacent pins designed with matched impedance). For the pin pairings available for differential conversions for your device, see the Chip Configuration details.

NOTE

Some input channels are from on-chip sources such as temperature sensors and reference voltage sources and may only be connected to individual instances of ADC. Some input channel options in the field descriptions may not be available for your device. See the chip-specific information for the channels supported on this device.

Diagram



Fields

Field	Function
31-22 —	Reserved
21 ALTBEN	Alternate Channel B Select Enable Enables the ALTB_ADCH to select the input for Channel B independent of ADCH register settings when CTYPE is configured to 01b or 11b. **NOTE** Setting CTYPE to differential (10b) still works but is not recommended. 0b - ALTBEN_ADCH disabled. Channel-A and Channel-B inputs are selected based on ADCH settings. 1b - ALTBEN_ADCH enabled. Channel-A inputs are selected by ADCH setting and Channel-B inputs are selected by ALTB_ADCH setting.
20-16 ALTB_ADCH	Alternate Channel B Input Channel Select When ALTBEN is set, ALTB_ADCH selects the Channel B input when CTYPE is configured to 01b or 11b. **NOTE** Some input channels are from internal resources such as temperature sensors and band gap voltage sources and may only be connected to individual instances of ADC. Some input channel options in field descriptions might not be available for your device. See the chip-specific information for the ADC channel assignments for your device. 0_0000b - Select CH0B 0_0001b - Select CH1B 0_0010b - Select CH2B 0_0011b - Select CH3B 0_0100b-1_1101b - Select corresponding channel CHnB 1_1110b - Select CH30B 1_1111b - Select CH31B
15-8 —	Reserved
7 MODE	Select Resolution of Conversions Selects the ADC resolution. 0b - Standard resolution. Single-ended 12-bit conversion; differential 13-bit conversion with 2's complement output. 1b - High resolution. Single-ended 16-bit conversion; differential 16-bit conversion with 2's complement output.
6-5	Conversion Type

Table continues on the next page...

Table continued from the previous page...

Field	Function
CTYPE	Chooses how a pair of A and B channels are converted. 00b - Single-Ended mode. Only A-side channel is converted. 01b - Single-Ended mode. Only B-side channel is converted. 10b - Differential mode. A-B. 11b - Dual-Single-Ended mode. Both A-side and B-side channels are converted independently.
4-0 ADCH	Input Channel Select Selects the input from one of the channel inputs or one of the input pairs, with CMDLn[CTYPE]. Each ADCH channel selection has an associated A-side and B-side input. **NOTE** Not all pairs should be converted as differential channels. See the chip-specific information for the pin pairings available for differential conversions for your device. **NOTE** Some input channels are from internal resources such as temperature sensors and band gap voltage sources and may only be connected to individual instances of ADC. Some input channel options in field descriptions might not be available for your device. See the chip-specific information for the ADC channel assignments for your device. 0_0000b - CH0A or CH0B or CH0A/CH0B pair. 0_0001b - CH1A or CH1B or CH1A/CH1B pair. 0_0010b - CH2A or CH2B or CH2A/CH2B pair. 0_0011b - CH3A or CH3B or CH3A/CH3B pair. 0_0100b-1_1101b - Select corresponding channel CHnA or CHnB or CHnA/CHnB pair. 1_1110b - CH30A or CH30B or CH30A/CH30B pair. 1_1111b - CH31A or CH31B or CH31A/CH31B pair.

41.6.18 Command High Buffer Register (CMDH1 - CMDH15)

Offset

For a = 1 to 15:

Register	Offset
CMDHa	FCh + (a × 8h)

Function

Controls channel selection and conversion options. There are 15 command buffers (CMDn), each constructed from two 32-bit registers (CMDHn:CMDLn) that can be configured for different channel selection and conversion options. Any command buffer is selected and used as the controlling command by association with a trigger event via configuration of TCTRLn[TCMD]. When ADC is executing commands, only one of the CMD buffers controls ADC conversions. Do not update the controlling CMD buffer while ADC is active. A write to a CMD buffer while that CMD buffer controls the ADC operation may cause unpredictable behavior.

NOTE

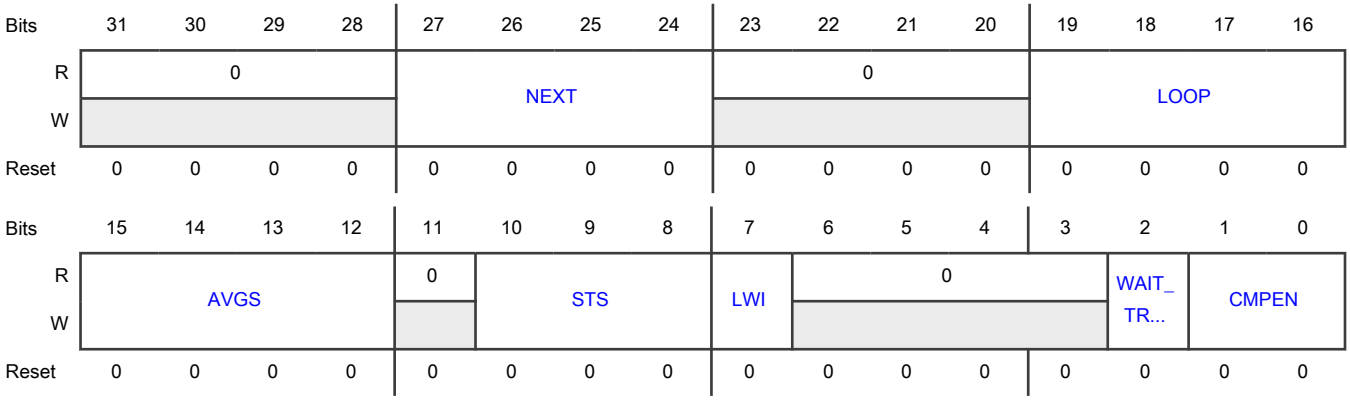
The 15 command buffers are numbered [CMDH1:CMDL1] through [CMDH15:CMDL15]. In NXP-supplied header files, these buffers are likely to be defined as two 15-element arrays that are indexed from 0. For example, the type declaration would be:

```
unsigned int CMDH[15];

unsigned int CMDL[15];
```

and software would access ADC0 CMDH1 as `ADC0->CMDH[0]` and ADC0 CMDL15 as `ADC0->CMDL[14]`.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 NEXT	Next Command Select Selects the next command to be executed after this command completes. Multiple commands can be configured in a scan configuration by linking the next command in a daisy-chain sequence. The command buffer number is not indicative of any particular order. The order of execution is strictly controlled by this field. For example, a sequence of commands could be CMD2 to CMD1 to CMD3. Unending circular command execution can be configured by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Setting the next command to 0h causes conversions to terminate at the completion of the command. Lower-priority trigger events cannot be serviced until a higher-priority triggered command (or sequence of commands) completes. 0000b - No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger. 0001b - CMD1 0010b-1110b - Select corresponding CMD command buffer register as next command 1111b - CMD15
23-20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
19-16 LOOP	Loop Count Select Selects the number of times that this command executes (and stores conversion result to RESFIFO) before finishing and transitioning to the next command or idle state. CMDHn[LWI] controls whether a single channel is converted on each iteration or an automatic channel increment results in channel scanning functionality. 0000b - Looping not enabled. Command executes one time. 0001b - Loop one time. Command executes two times. 0010b - Loop two times. Command executes three times. 0011b-1110b - Loop corresponding number of times. Command executes LOOP + 1 times. 1111b - Loop 15 times. Command executes 16 times.
15-12 AVGS	Hardware Average Select Selects how many ADC conversions are averaged to create the ADC result (2^{AVGS}). An internal storage buffer captures temporary results while the averaging iterations are executed. Hardware averaging is a nested loop control and does not extend across LOOP boundaries. See Functional description for usage of AVGS, LOOP, and NEXT fields in command execution sequencing. 0000b - Single conversion 0001b - 2 0010b - 4 0011b - 8 0100b - 16 0101b - 32 0110b - 64 0111b - 128 1000b - 256 1001b - 512 1010b - 1024
11 —	Reserved
10-8 STS	Sample Time Select Selects the total sampling time. When this field contains a non-zero value, the sample time is ($3.5 + 2^{STS}$) ADCK cycles. The shortest sample time maximizes conversion speed for lower-impedance inputs. Extending sample time allows higher-impedance inputs to be sampled accurately. Longer sample times can lower overall power consumption when command looping and sequencing are configured and high conversion rates are not required. 000b - Minimum sample time of 3.5 ADCK cycles.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001b - 5.5 ADCK cycles. (3.5 + 2^1 ADCK cycles) 010b - 7.5 ADCK cycles. (3.5 + 2^2 ADCK cycles) 011b - 11.5 ADCK cycles. (3.5 + ^{3} ADCK cycles) 100b - 19.5 ADCK cycles. (3.5 + 2^4 ADCK cycles) 101b - 35.5 ADCK cycles. (3.5 + 2^5 ADCK cycles) 110b - 67.5 ADCK cycles. (3.5 + 2^6 ADCK cycles) 111b - 131.5 ADCK cycles. (3.5 + 2^7 ADCK cycles)
7 LWI	Loop with Increment Enables automatic channel incrementing. When 0, the LOOP field selects the number of times the selected channel is converted consecutively. When 1, automatic channel incrementing is enabled and the LOOP field defines how many consecutive channels are converted as part of the command execution. Example 1: LOOP = 8h, LWI = 0b, CMDLn[CTYPE] = 0h, CMDLn[ADCH] = Dh. Convert on channel 13A 9 times. Example 2: LOOP = 8h, LWI = 1b, CMDLn[CTYPE] = 0h, CMDLn[ADCH] = Dh. Run channels 13A-21A each one time. Maximum channel scanning using a single command buffer is defined by the maximum value of the LOOP field (16). 0b - Disabled 1b - Enabled
6-3 —	Reserved
2 WAIT_TRIG	Wait for Trigger Assertion Before Execution Selects whether commands execute automatically or a trigger must be received before execution. When 0, wait states are added before the command until the active trigger is asserted again. When WAIT_TRIG = 0, each command is automatically executed when called. 0b - Command executes automatically. 1b - Active trigger must be asserted again before executing this command.
1-0 CMPEN	Compare Function Enable Enables the automatic compare function. Selects whether to store only when the comparison operation is true, following ADC channel input sampling, conversion, and averaging. When the compare function is enabled, the conversion result is compared to the compare value registers (CVn[CVH] and CVn[CVL]). See Compare Function for details about the options for command sequencing related to the compare function. 00b - Disabled 01b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Enabled. Store on true. 11b - Enabled. Repeat channel acquisition (sample, convert, and compare) until true.

41.6.19 Compare Value Register (CV1 - CV15)

Offset

For a = 1 to 15:

Register	Offset
CVa	1FCh + (a × 4h)

Function

Contains values used to compare the conversion result when the compare function is enabled. This register is formatted like the D field in Data Result FIFO Register (RESFIFO0 - RESFIFO1), which has in different definitions for bit position and value format in different modes. There is a direct association of each compare value register to a specific command buffer register. For example, CV1 is only used during execution of CMD1 command.

When ADC is executing commands, do not update the CVn register associated with the active command (CMDn). Writes to associated CVn register during this time may result in unpredictable behavior.

NOTE

The 15 compare value registers are numbered [CV1] through [CV15]. In NXP-supplied header files, these registers are likely to be defined as a 15-element array indexed from 0. For example, the type declaration would be:

```
unsigned int CV[15];
```

and software would access ADC0 CV1 as `ADC0->CV[0]` and ADC0 CV15 as `ADC0->CV[14]`.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CVH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CVL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 CVH	Compare Value High Determines the high compare value. The compare function can be configured to check whether the result: • Is less than comparison values • Is greater than comparison values • Falls within a range of two comparison values • Falls outside a range of two comparison values. After the input is sampled and converted and any averaging iterations are performed, CVL and CVH can be used in a compare operation on the result. See Compare Function for a description of CVH usage.
15-0 CVL	Compare Value Low Determines the low compare value. The compare function can be configured to check whether the result: • Is less than comparison values • Is greater than comparison values • Falls within a range of two comparison values • Falls outside a range of two comparison values. After the input is sampled and converted and any averaging iterations are performed, CVL and CVH can be used in a compare operation on the result. See Compare Function for a description of CVL usage.

41.6.20 Data Result FIFO Register (RESFIFO0 - RESFIFO1)

Offset

Register	Offset
RESFIFO0	300h
RESFIFO1	304h

Function

Stores the data result of ADC conversions in a 16-entry FIFO. Several tag fields of source command and trigger information are stored with the data. FCTRLn[FCOUNT] indicates how many valid data words are stored in the RESFIFO. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements FCTRLn[FCOUNT]. The FIFO can be emptied by successive reads of RESFIFO. The FIFO is reset by writing 0b1 to CTRL[RSTFIFOn].

The following table describes the format of data in the result FIFO in different modes of operation. The sign bit is the MSB in signed 2's complement modes. For example, when configured for 12-bit single-ended mode, D[15] and D[2:0] become 0. When configured for 13-bit differential mode, D[15] is the sign bit, and D[2:0] becomes 0.

Table 369. Data result register format description

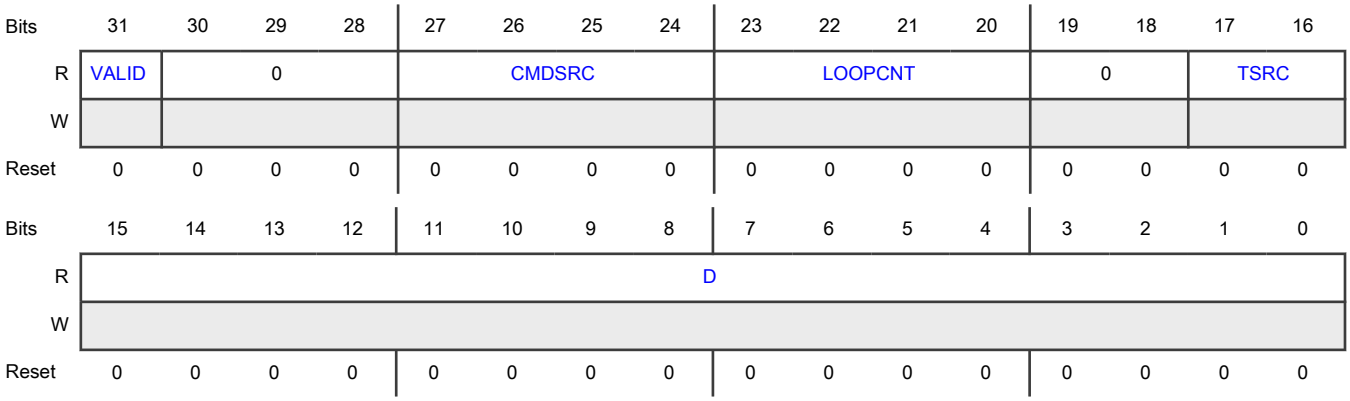
Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
16-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Signed 2's complement
16-bit single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned, 16-bit magnitude
13-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Signed 2's complement, left justified, zero extended
12-bit single-ended	0	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Unsigned, zero in D[15] and D[2:0]

NOTE

S: Sign bit;

D: Data, 2's complement data when indicated

Diagram



Fields

Field	Function
31 VALID	FIFO Entry is Valid Indicates whether the FIFO entry is valid, which determines what happens to reads from RESFIFO. 0b - FIFO is empty. Discard any read from RESFIFO. 1b - FIFO contains data. FIFO record read from RESFIFO is valid.
30-28 —	Reserved
27-24	Command Buffer Source

Table continues on the next page...

Table continued from the previous page...

Field	Function
CMDSRC	Indicates the executed command buffer that generated this result. 0000b - Not a valid value CMDSRC value for a data word in RESFIFO. 0h is only found in the initial FIFO state, prior to the storage of an ADC conversion result into a RESFIFO buffer. 0001b - CMD1 0010b-1110b - Corresponding command buffer used as control settings for this conversion. 1111b - CMD15
23-20 LOOPCNT	Loop Count Value Indicates the loop count value during the command that generated this result. When CMDHn[LOOP] is non-zero, results are stored multiple times during command execution at the loop boundary. 0000b - Result is from initial conversion in command. 0001b - Result is from second conversion in command. 0010b-1110b - Result is from (LOOPCNT + 1) conversion in command. 1111b - Result is from 16th conversion in command.
19-18 —	Reserved
17-16 TSRC	Trigger Source Indicates the trigger source that initiated a conversion and generated this result. When multiple commands are chained together using CMDHn[NEXT], this field indicates the trigger source that started the command sequence. 00b - Trigger source 0 01b - Trigger source 1 10b - Trigger source 2 11b - Trigger source 3
15-0 D	Data Result Contains the result of an ADC conversion. The formatting for the data in D is summarized in Table 369.

41.6.21 Calibration General A-Side Registers (CAL_GAR0)

Offset

Register	Offset
CAL_GAR0	400h

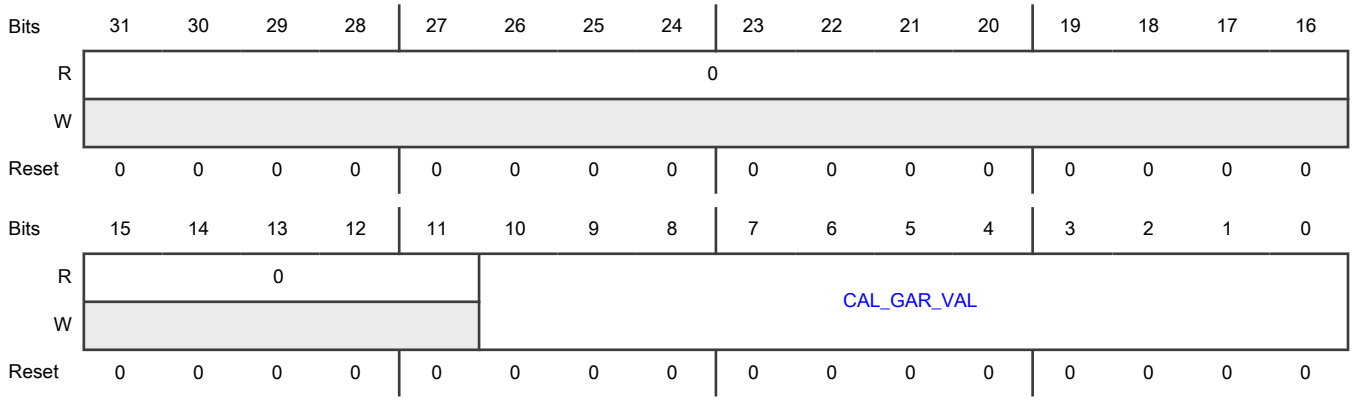
Function

Corrects for linearity errors of the A-side converter. All 33 A-side general calibration value registers (CAL_GAR0-CAL_GAR32) contain calibration information that is automatically updated during the self-calibration sequence. See Calibration functions for more information on completing ADC calibration steps.

The calibration values in the CAL_GAR registers affect the conversion result by conditionally being subtracted from the conversion before the result is transferred into the FIFOs. Calibration must be run each time ADC is powered down or a hard reset is issued. To reduce the latency required to run calibration, the CAL_GAR values can be stored in non-volatile memory after an initial calibration. These values can be recovered prior to the first ADC conversion. If these registers are set to values not generated by the calibration function, the linearity error specifications may not be met.

The CAL_GAR registers are only read-and-write accessible when ADC is disabled with CTRL[ADCEN] = 0. Access time when writing to these registers is larger than three ADC clock cycles. Wait states are inserted on the bus to meet synchronization timing to the associated CAL_GAR register. The width of each register in this array is non-uniform. The exact width of each register is summarized in Calibration General A-Side and B-Side Widths.

Diagram



Fields

Field	Function
31-11 —	Reserved
10-0 CAL_GAR_VAL	Calibration General A Side Register Element

41.6.22 Calibration General A-Side Registers (CAL_GAR1)

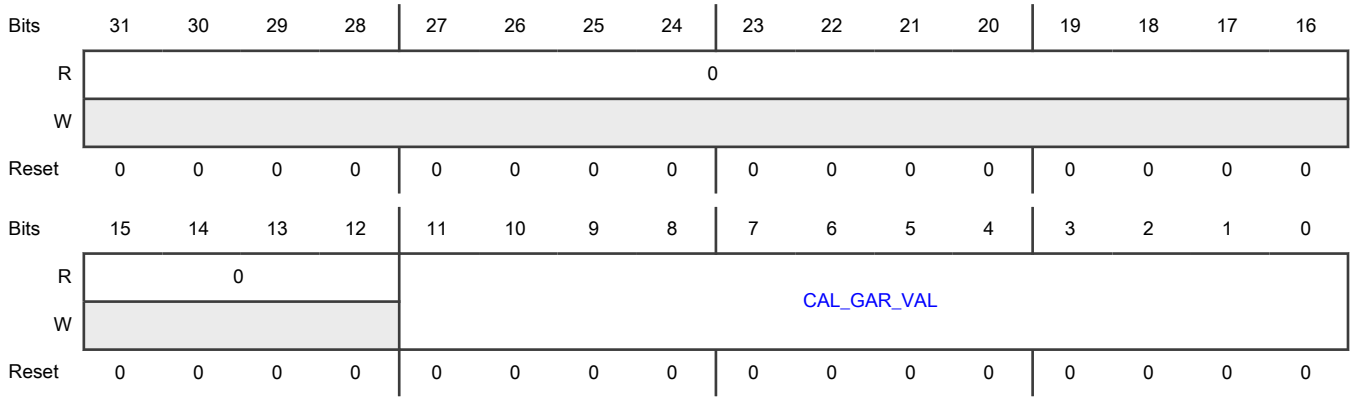
Offset

Register	Offset
CAL_GAR1	404h

Function

Calibration register CAL_GAR1. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Diagram



Fields

Field	Function
31-12 —	Reserved
11-0 CAL_GAR_VAL	Calibration General A Side Register Element

41.6.23 Calibration General A-Side Registers (CAL_GAR2 - CAL_GAR3)

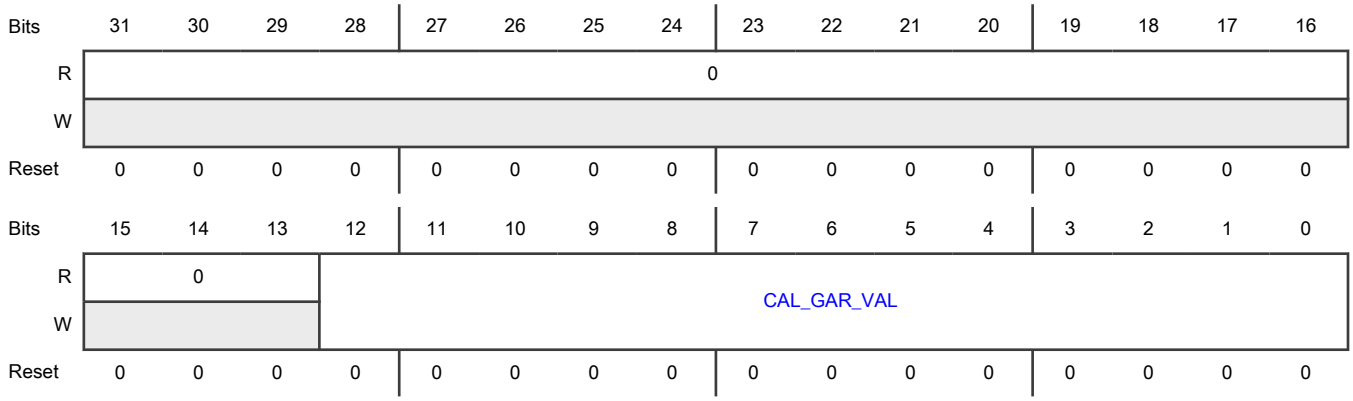
Offset

Register	Offset
CAL_GAR2	408h
CAL_GAR3	40Ch

Function

Calibration registers CAL_GAR2 and CAL_GAR3. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Diagram



Fields

Field	Function
31-13 —	Reserved
12-0 CAL_GAR_VAL	Calibration General A Side Register Element

41.6.24 Calibration General A-Side Registers (CAL_GAR4 - CAL_GAR7)

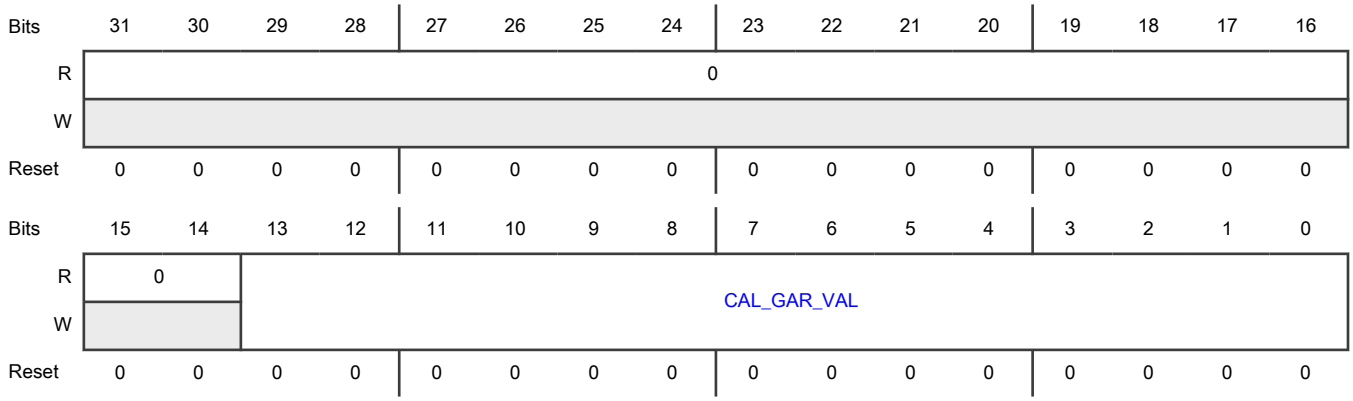
Offset

Register	Offset
CAL_GAR4	410h
CAL_GAR5	414h
CAL_GAR6	418h
CAL_GAR7	41Ch

Function

Calibration registers CAL_GAR4 through CAL_GAR7. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Diagram



Fields

Field	Function
31-14 —	Reserved
13-0	Calibration General A Side Register Element

Table continues on the next page...

Table continued from the previous page...

Field	Function
CAL_GAR_VAL	

41.6.25 Calibration General A-Side Registers (CAL_GAR8 - CAL_GAR15)

Offset

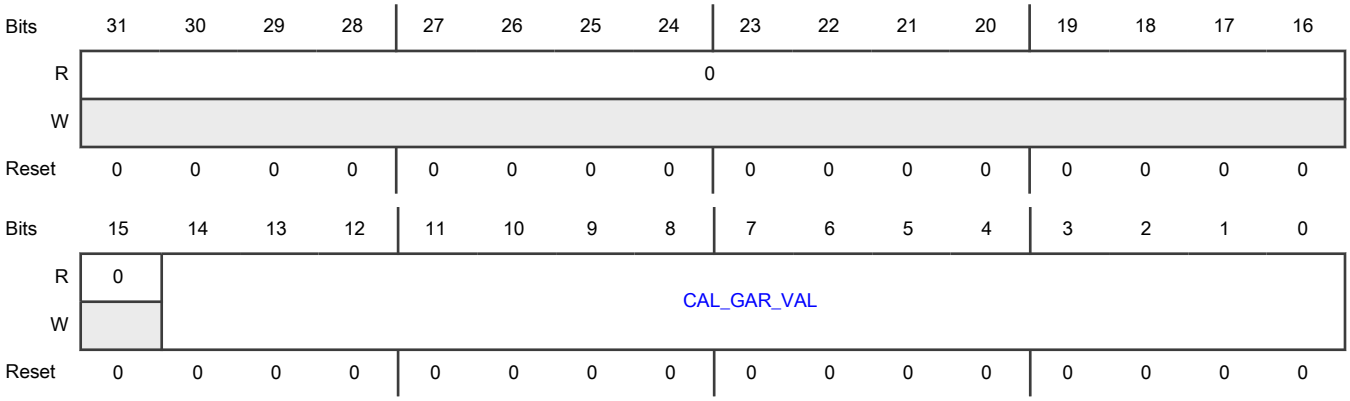
For a = 8 to 15:

Register	Offset
CAL_GARa	400h + (a × 4h)

Function

Calibration registers CAL_GAR8 through CAL_GAR15. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Diagram



Fields

Field	Function
31-15 —	Reserved
14-0 CAL_GAR_VAL	Calibration General A Side Register Element

41.6.26 Calibration General A-Side Registers (CAL_GAR16 - CAL_GAR31)

Offset

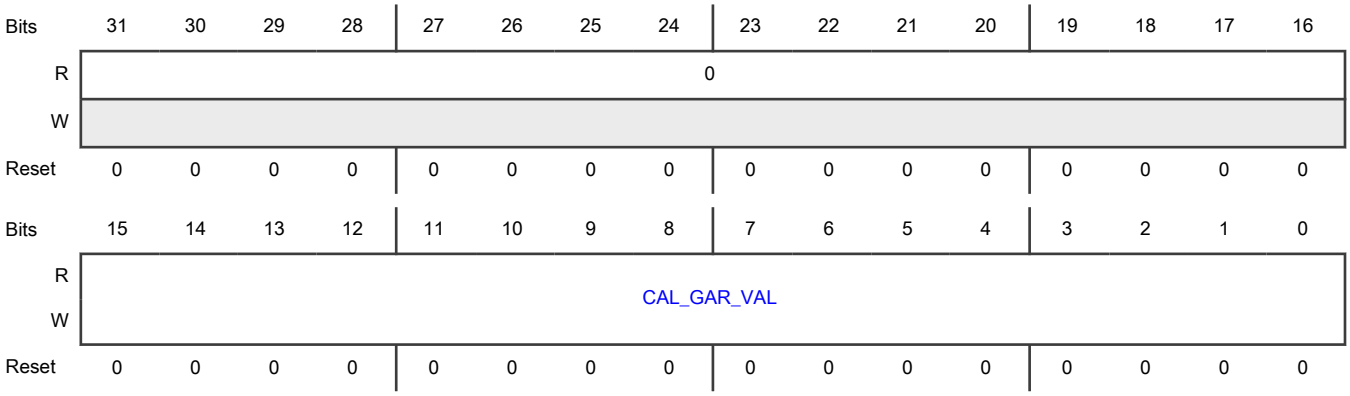
For a = 16 to 31:

Register	Offset
CAL_GARa	400h + (a × 4h)

Function

Calibration registers CAL_GAR16 through CAL_GAR31. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CAL_GAR_VAL	Calibration General A Side Register Element

41.6.27 Calibration General A-Side Registers (CAL_GAR32)

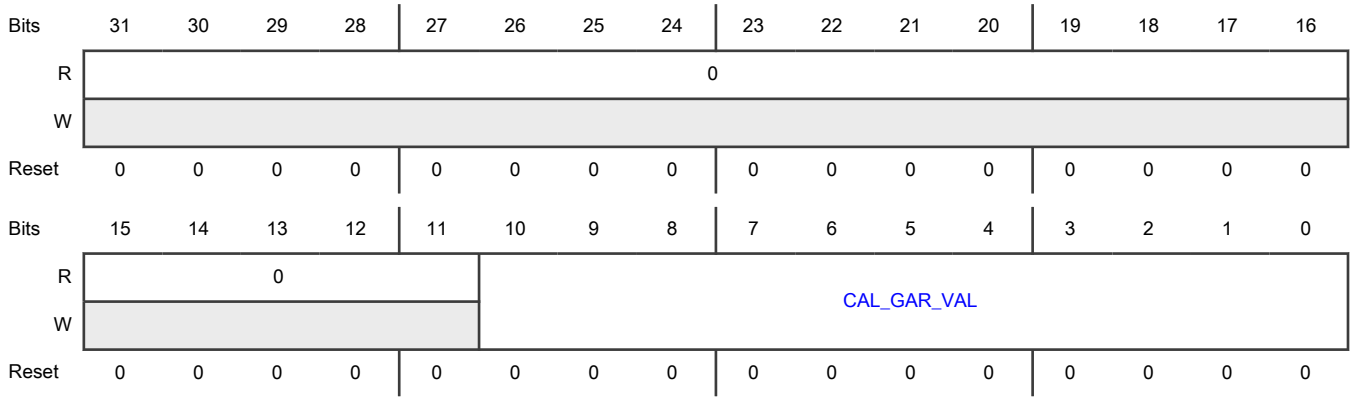
Offset

Register	Offset
CAL_GAR32	480h

Function

Calibration register CAL_GAR32. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Diagram



Fields

Field	Function
31-11 —	Reserved
10-0 CAL_GAR_VAL	Calibration General A Side Register Element

41.6.28 Calibration General B-Side Registers (CAL_GBR0)

Offset

Register	Offset
CAL_GBR0	500h

Function

Corrects for linearity errors of the B-side converter. All 33 B-side general calibration value registers (CAL_GBR0-CAL_GBR32) contain calibration information that is automatically updated during the self-calibration sequence. See Calibration functions for more information on completing ADC calibration steps.

The calibration values in the CAL_GBR registers affect the conversion result by conditionally being subtracted from the conversion before the result is transferred into the FIFOs. Calibration must be run each time ADC is powered down or a hard reset is issued. To reduce the latency required to run calibration, the CAL_GBR values can be stored in non-volatile memory after an initial calibration. These values can be recovered prior to the first ADC conversion. If these registers are set to values not generated by the calibration function, the linearity error specifications may not be met.

The CAL_GBR registers are only read-and-write accessible when ADC is disabled with CTRL[ADCEN] = 0. Access time when writing to these registers is larger than three ADC clock cycles. Wait states are inserted on the bus to meet synchronization timing to the associated CAL_GBR register. The width of each register in this array is non-uniform. The exact width of each register is summarized in Calibration General A-Side and B-Side Widths.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				CAL_GBR_VAL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-11 —	Reserved
10-0 CAL_GBR_VAL	Calibration General B Side Register Element

41.6.29 Calibration General B-Side Registers (CAL_GBR1)

Offset

Register	Offset
CAL_GBR1	504h

Function

Calibration register CAL_GBR1. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				CAL_GBR_VAL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11-0 CAL_GBR_VAL	Calibration General B Side Register Element

41.6.30 Calibration General B-Side Registers (CAL_GBR2 - CAL_GBR3)

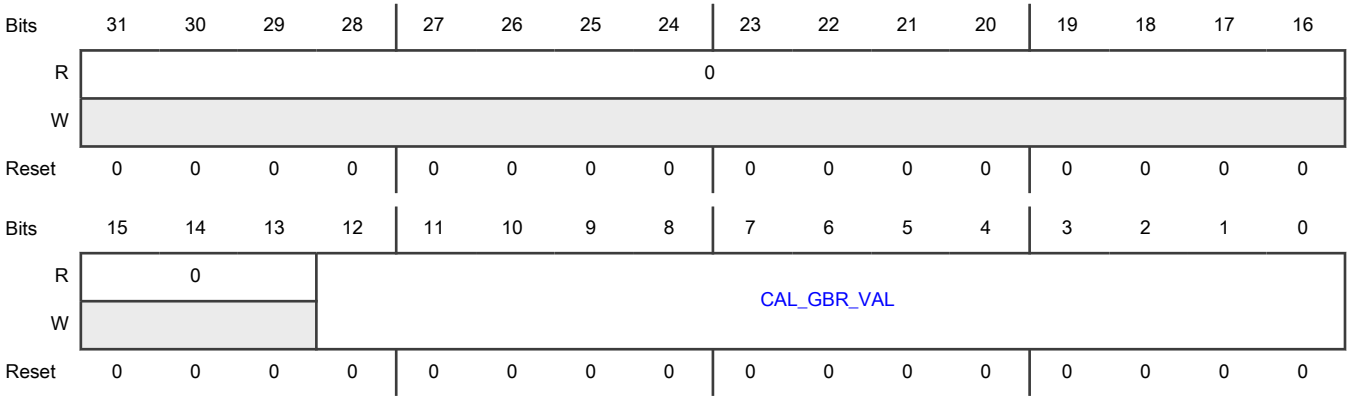
Offset

Register	Offset
CAL_GBR2	508h
CAL_GBR3	50Ch

Function

Calibration registers CAL_GBR2 and CAL_GBR3. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Diagram



Fields

Field	Function
31-13 —	Reserved
12-0 CAL_GBR_VAL	Calibration General B Side Register Element

41.6.31 Calibration General B-Side Registers (CAL_GBR4 - CAL_GBR7)

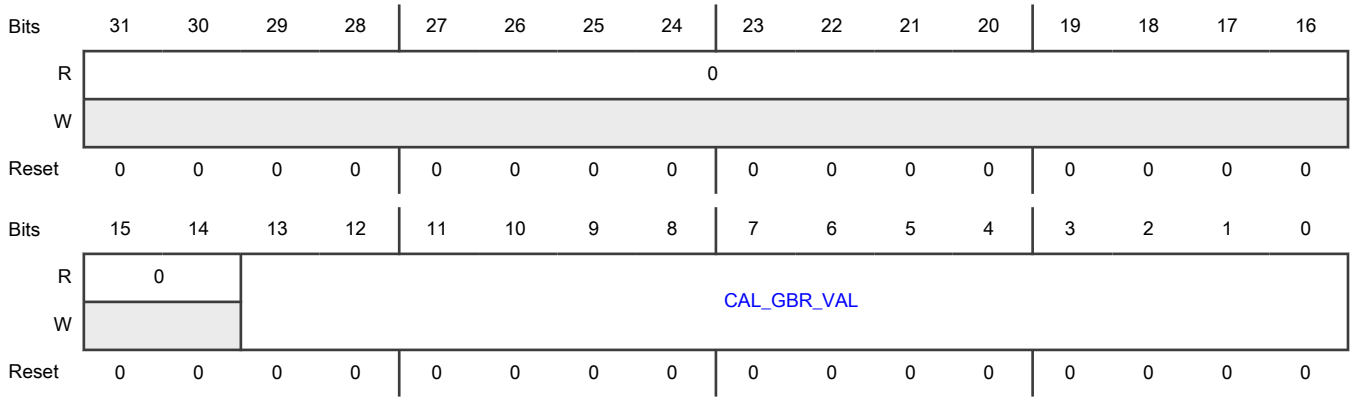
Offset

Register	Offset
CAL_GBR4	510h
CAL_GBR5	514h
CAL_GBR6	518h
CAL_GBR7	51Ch

Function

Calibration registers CAL_GBR4 through CAL_GBR7. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Diagram



Fields

Field	Function
31-14 —	Reserved
13-0 CAL_GBR_VAL	Calibration General B Side Register Element

41.6.32 Calibration General B-Side Registers (CAL_GBR8 - CAL_GBR15)

Offset

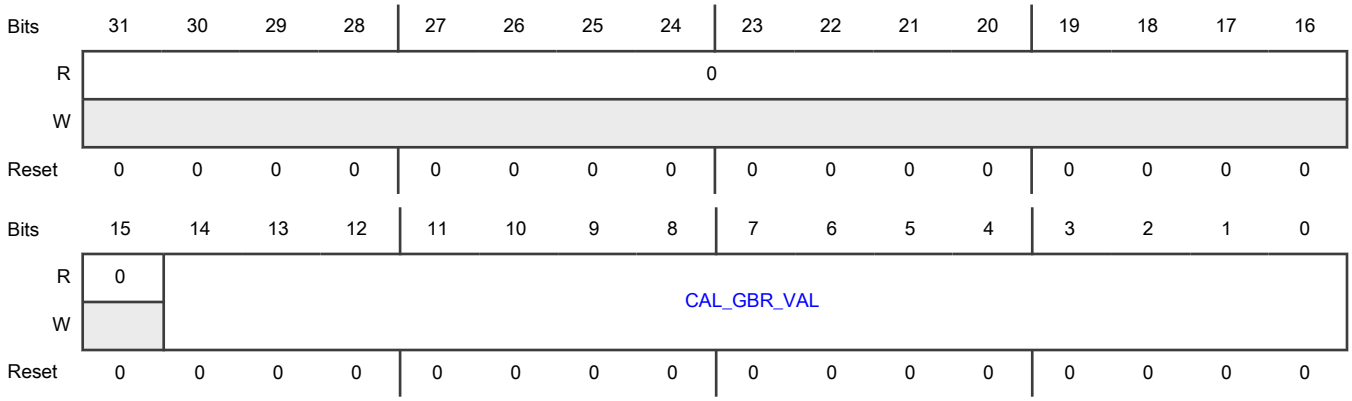
For a = 8 to 15:

Register	Offset
CAL_GBRa	500h + (a × 4h)

Function

Calibration registers CAL_GBR8 through CAL_GBR15. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Diagram



Fields

Field	Function
31-15 —	Reserved
14-0 CAL_GBR_VAL	Calibration General B Side Register Element

41.6.33 Calibration General B-Side Registers (CAL_GBR16 - CAL_GBR31)

Offset

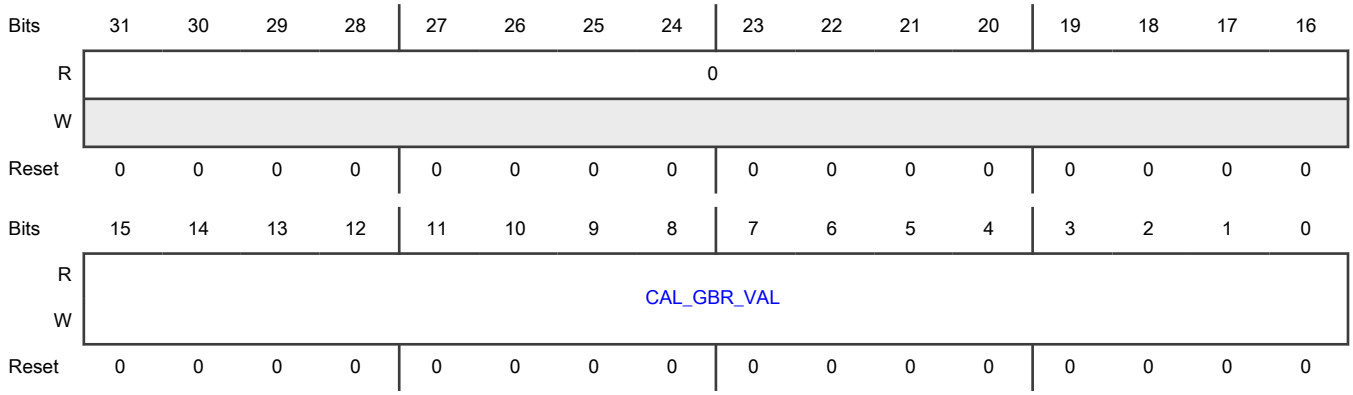
For a = 16 to 31:

Register	Offset
CAL_GBRa	500h + (a × 4h)

Function

Calibration registers CAL_GBR16 through CAL_GBR31. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CAL_GBR_VAL	Calibration General B Side Register Element

41.6.34 Calibration General B-Side Registers (CAL_GBR32)

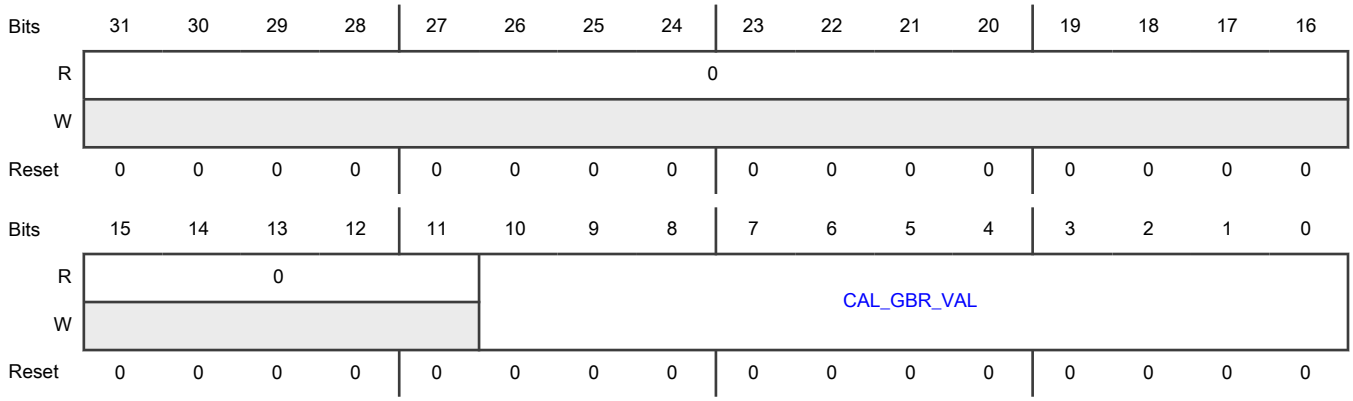
Offset

Register	Offset
CAL_GBR32	580h

Function

Calibration register CAL_GBR32. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Diagram



Fields

Field	Function
31-11 —	Reserved
10-0 CAL_GBR_VAL	Calibration General B Side Register Element

Chapter 42
12-bit Digital-to-Analog Converter (DAC)

42.1 Chip-specific 12-bit DAC information

Table 370. Reference links to related information

Topic	Related module	Reference
Full description	12-bit DAC	12-bit Digital-to-Analog Converter (DAC)
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See DAC0_TRIG and DAC1_TRIG registers in INPUTMUX

42.1.1 Module instances

This device has two instances of the 12-bit 1Msps DAC module, DAC0, DAC1, with compatible DAC Controller. Set controller FIFO size to 4 (16 words).

42.1.2 12-bit DAC references

- VREFH0 = VDD_ANA
- VREFH1 = VREFO
- VREFH2 = VREFH

42.1.3 12-bit DAC chip-specific initialization

To initialize the 12-bit DAC module:

1. Set the clock divider for the DAC with DACxCLKDIV[DIV] bit in SYSCON register.
2. Enable the clock to the DAC (AHBCLKCTRLn[DACn] = 1) in SYSCON registers. This enables the register interface and the peripheral function clock.
3. Power up the DAC with ACTIVE_CFG1 and LP_CFG1 registers in SPC. See SOC_CNTRL fields.
4. Reset and release the DAC peripheral reset by setting the PRESETCTRLSETn[DACn_RST] bit and then set PRESETCTRLCLRn[DACn_RST] bit.
5. Reset the DAC logic by setting and clearing bit RCR[SWRST]. See Reset Control (RCR).
6. Reset the DAC FIFO by setting and clearing bit RCR[FIFORST].
7. Enable VREF module SOC_CNTRL[VREF_EN] = 1 in ACTIVE_CFG1 register in SPC module. See SOC_CNTRL fields. DAC PTAT and ZTC current sources come from VREF module.
8. If needed, initialize the DAC reference source before selecting the DAC reference with GCR[DACRFS]. See Global Control (GCR)
   - If using VREF_OUT as reference, see Chip-specific VREF information link to VREF_OUT initialization
9. Configure the DAC as described in Initialization.

42.1.4 12-bit DAC trigger inputs

12-bit DAC trigger sources get routed through the INPUTMUX. See the registers DAC0_TRIG and DAC1_TRIG in the INPUTMUX chapter for available trigger sources.

42.2 Overview

The 12-bit general-purpose digital-to-analog converter is a Low-power DAC. The output of the DAC can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.

42.2.1 Block diagram

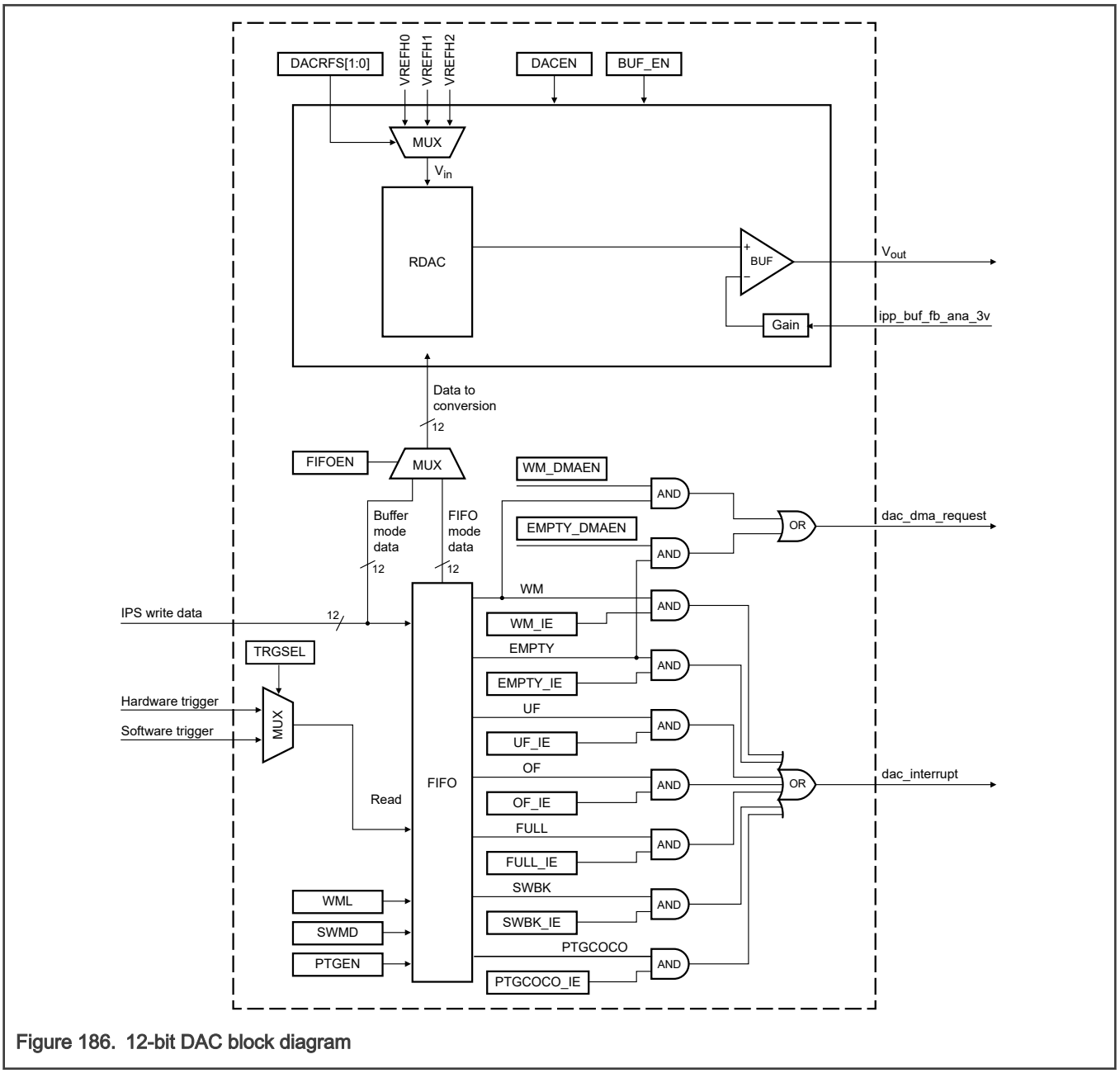


Figure 186. 12-bit DAC block diagram

42.2.2 Features

- Includes an on-chip programmable reference generator output. The voltage output ranges from 1/4096 V_{in} to V_{in}, and the step is 1/4096 V_{in}, where V_{in} is the input reference voltage.
- Selects V_{in} from three reference sources.
- Supports 16-word depth FIFO with the configurable watermark.
- Uses multiple operation modes.
  - Buffer mode
  - FIFO mode
  - Swing Back mode
  - Periodic Trigger mode
- Supports software trigger and hardware trigger.
- Provides selectable performance levels: Low-Power mode(LP) and Lower Low-Power mode(LLP).
- Provides interrupt and DMA support.

42.3 Functional description

12-bit DAC can select one of the three reference inputs: VREFH0, VREFH1, and VREFH2 as the reference voltage, by DACRFS. See the chip-specific 12-bit DAC information to determine the source options for VREFH0, VREFH1, and VREFH2.

12-bit DAC converts a 12-bit data to a stepped analog output voltage when enabled. The output voltage range is from V_{in} to V_{in}/4096, and the step is V_{in}/4096.

You can configure the 12-bit DAC to Buffer mode, FIFO mode, Swing Back mode, or Periodic Trigger mode.

NOTE

You must configure 12-bit DAC before any conversion. During conversion or FIFO write operation, you must not change the configuration or disable DACEN. If you switch to another mode, you must first write a 1 to SWRST to reset 12-bit DAC or write 0 to DACEN.

42.3.1 Buffer mode

You can enable Buffer mode with DACEN.

In Buffer mode, any write to DATA pushes the data to the analog conversion without trigger support.

NOTE

The data write speed must not be higher than the analog conversion speed.

42.3.2 FIFO mode

You can enable FIFO mode with FIFOEN and DACEN.

In FIFO mode, for a 32-/16-bit write to DATA, put the data into the FIFO and the write pointer increments automatically. For an 8-bit write, only if you write both the bytes, the write pointer increases. You must ensure that an 8-bit write is done in pair and write both the upper and lower bytes. There is no priority, which byte to write first.

When a software or hardware trigger comes, the read pointer pointed data in FIFO pushes to the analog conversion, and the read pointer increments automatically. If the trigger comes when FIFO is empty, the data to analog remains unchanged, and an underflow error flag sets.

42.3.3 Swing Back mode

You can enable Swing Back mode with FIFOEN, SWMD, and DACEN.

The read pointer swings between the writer pointer and zero in Swing Back mode. The trigger increases the read pointer till the writer pointer and decreases the read pointer till zero, and so on. EMPTY, FULL, and WM have no meaning and you should ignore them during Swing Back mode.

NOTE

In Swing Back mode, the FIFO should first write at least two data; otherwise, this mode will not work correctly. You should not write the FIFO when Swing Back mode is in progress.

42.3.4 Periodic Trigger mode

You can enable Periodic Trigger mode with FIFOEN, PTGEN, and DACEN.

Due to the trigger (hardware or software) synchronization logic, there is a delay time variation between each conversion. If there is a need to trigger DAC periodically, NXP recommends you to use Periodic Trigger mode.

In Periodic Trigger mode, you only send the first trigger. Then after every [PTG_PERIOD+1] read clock (RCLK) cycle, an internal trigger automatically triggers the DAC. There are [PTG_NUM] internal triggers, therefore, in total [PTG_NUM+1] conversions, including the first trigger that you send. Write 0 to PTGEN to terminate the current conversion queue. Then, after the current conversion completes, the conversion terminates, and the PTGCOCO sets. If PTG_NUM becomes zero, the infinite triggers follow the first hardware/software trigger until you write 0 to PTGEN. In any case, FIFORST or SWRST can terminate the conversion.

During the periodic trigger mode conversion, you can ignore the additional hardware/software triggers. You can start another queue after PTGCOCO becomes 1.

During Periodic Trigger Mode, WM feeds data to FIFO to avoid underflow.

You can combine Periodic Trigger mode with SWMD to periodically trigger DAC and swing back.

42.3.5 Low-power mode operation

If the RCLK remains enabled, 12-bit DAC can be functional with FIFO mode or Swing Back mode or Periodic Trigger mode during Stop and Wait modes, using the hardware trigger initiating the conversion.

42.3.6 DMA

After the corresponding DMA enable register field becomes 1, you can automatically generate DMA requests when WM or EMPTY becomes 1. When you enable DMA, the error flags can only generate an interrupt.

42.3.7 Clocks

12-bit DAC block requires the following clock(s):

- Bus clock for register access and writing data to FIFO or Buffer.
- Read clock to push data from FIFO or Buffer to the analog DAC.

42.3.8 Resets

The following are the 12-bit DAC resets:

1. Global reset fed into the block. This global reset resets everything.
2. SWRST: resets all 12-bit DAC registers and internal logic.
3. FIFORST: resets the FIFO pointers in DAC FIFO Pointer (FPR) and flags in FIFO Status (FSR).

42.3.9 Interrupts

After the corresponding interrupt enable register field becomes 1, warning flags and error flags can generate interrupt. The warning flags are EMPTY, FULL, WM, SWBK, and PTGCOCO. The error flags are OF and UF.

42.4 DAC external signal descriptions

Table 371. DAC external signal descriptions

Signal	Description	I/O
VREFH0	DAC reference high voltage 0	I
VREFH1	DAC reference high voltage 1	I
VREFH2	DAC reference high voltage 2	I
Vout	DAC output	O

42.5 Initialization

To initialize 12-bit DAC:

1. Configure the read clock at chip level.
2. Configure Work mode and GCR enables 12-bit DAC.
3. Configure Interrupt Enable (IER) if required.
4. Configure DMA Enable (DER) if required.
5. Configure Periodic Trigger Control (PCR) if required.
6. Write data to DATA to fill the FIFO, if FIFO mode.
7. Wait analog start-up time
8. Use hardware or software trigger to push data to the analog DAC to convert, if FIFO mode.
9. Write data directly to DATA to push data to the analog DAC to convert, if Buffer mode.

42.6 Application Information

Digital delay

Due to the resynchronization and data latch process, there is a delay between the trigger posedge and analog voltage settling. The delay is (2+ LATCH_CYC)*T to (3+ LATCH_CYC)*T, where T is the RCLK period, and LATCH_CYC*T should not be smaller than 40 ns. So the delay has one T uncertainty. The fast RCLK can achieve a small delay.

RCLK frequency

For discrete conversion, the RCLK frequency has no low limit. But slow RCLK involves large digital delay and uncertainty.

For continuous conversion, the RCLK frequency should be at least 5 times the trigger speed.

For continuous conversion, due to the one cycle uncertainty RCLK, the trigger period should not less than (1000ns + T). Where 1000 ns period means the maximum analog conversion speed 1 Msps.

To achieve the full speed of DAC analog, you must configure to the internal periodic trigger mode.

For Internal Periodic Trigger mode, RCLK frequency should be at least two times the conversion speed. For example, to achieve 1 Msps conversion speed, at least 2 MHz RCLK requires in Periodic Trigger mode.

42.7 Memory map and register definition

All registers are accessible in 8-, 16-, or 32-bit accesses. For efficiency, 32-bit accesses are recommended. Access to an address outside the valid memory map generates a bus error.

42.7.1 12-bit DAC register descriptions

Access to an address outside the valid memory map generates a bus error.

42.7.1.1 DAC memory map

DAC0 base address: 4010_F000h

DAC1 base address: 4011_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version Identifier (VERID)	32	R	0100_0000h
4h	Parameter (PARAM)	32	R	0000_0003h
8h	Data (DATA)	32	RW	0000_0000h
Ch	Global Control (GCR)	32	RW	0000_0100h
10h	DAC FIFO Control (FCR)	32	RW	0000_0000h
14h	DAC FIFO Pointer (FPR)	32	R	0000_0000h
18h	FIFO Status (FSR)	32	RW	0000_0002h
1Ch	Interrupt Enable (IER)	32	RW	0000_0000h
20h	DMA Enable (DER)	32	RW	0000_0000h
24h	Reset Control (RCR)	32	RW	0000_0000h
28h	Trigger Control (TCR)	32	RW	0000_0000h
2Ch	Periodic Trigger Control (PCR)	32	RW	0000_0000h

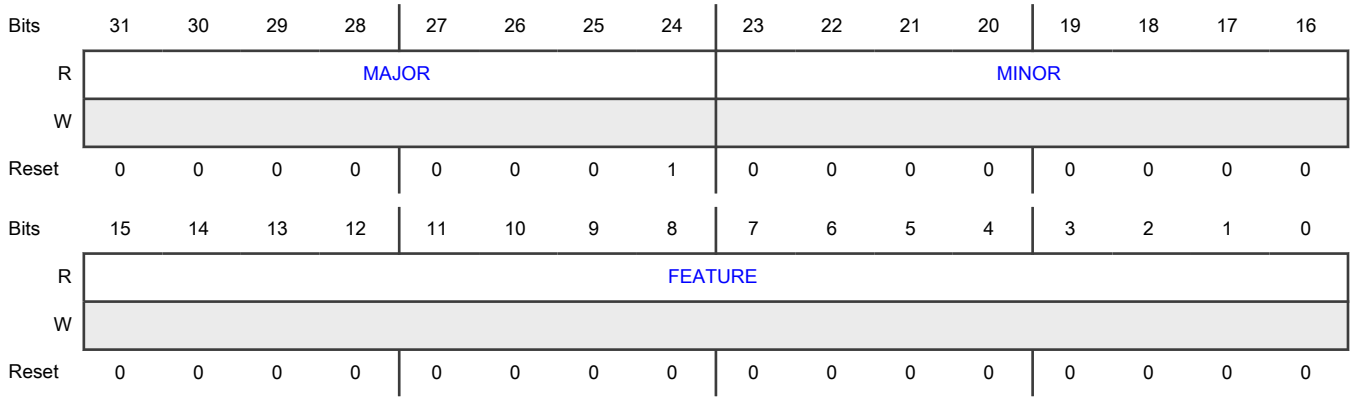
42.7.1.2 Version Identifier (VERID)

Offset

Register	Offset
VERID	0h

Function
Indicates the version integrated for this instance on the chip.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15-0 FEATURE	Feature Identification Number Returns the feature set number.

42.7.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function

Indicates the feature parameters for this instance on the chip.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													FIFOSZ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Fields

Field	Function
31-3 —	Reserved
2-0 FIFOSZ	FIFO Size Indicates that the FIFO depth is 2^(FIFOSZ+1) words. 000b - Reserved 001b - FIFO depth is 4 010b - FIFO depth is 8 011b - FIFO depth is 16 100b - FIFO depth is 32 101b - FIFO depth is 64 110b - FIFO depth is 128 111b - FIFO depth is 256

42.7.1.4 Data (DATA)

Offset

Register	Offset
DATA	8h

Function

Holds the FIFO entry in FIFO mode and serves as the buffer in Buffer mode.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0											
W					DATA											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11-0 DATA	FIFO Entry or Buffer Entry Specifies that in FIFO mode, Swing Back mode, or Periodic Trigger mode, this is the FIFO data entry. In Buffer mode, write to this field pushes the data to analog without trigger support.

42.7.1.5 Global Control (GCR)

Offset

Register	Offset
GCR	Ch

Function

Contains configuration options for DAC function, such as mode select, trigger select, and so on. It also includes the reference current select for the analog DAC.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								BUF_SPD...	0	IREF_ZT...	IREF_PT...	0		BUF_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				LATCH_CYC				0	PTGEN	TRGSEL	SWMD	FIFOEN	DACRFS		DACEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 BUF_SPD_CTRL	OPAMP as Buffer, Speed Control Signal **NOTE** See the DAC electrical characteristics of the chip datasheet for more information on the impact of the modes below. 0b - Lower Low-Power mode 1b - Low-Power mode
22 —	Reserved
21 IREF_ZTC_EXT_SEL	External On-Chip ZTC Current Reference Select Selects the external ZTC current reference, which is from the VREF module. See the Chip-specific section for more information. 0b - Not selected 1b - Selected
20 IREF_PTAT_EXT_SEL	External On-Chip PTAT Current Reference Select Selects the external PTAT current reference, which is from the VREF module. See the Chip-specific section for more information. 0b - Not selected 1b - Selected
19-18 —	Reserved
17 BUF_EN	Buffer Enable Uses OPAMP as the DAC analog buffer. 0b - Not used 1b - Used
16-12 —	Reserved
11-8 LATCH_CYC	RCLK Cycles Before Data Latch Configures the DAC sync cycles, which reduces the glitch on the output. The sync time is (LATCH_CYC+1) RCLK cycles. You must configure this register per the RCLK frequency. The recommended sync time is at least 40 ns.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The recommended LATCH_CYC configuration is: Sync time is (LATCH_CYC+1) RCLK cycles, when (25×LATCH_CYC) MHz < RCLK ≤ (25×LATCH_CYC+25) MHz. Example 1, if f_{RCLK} is 48 MHz (period is 20.8 ns), you can configure LATCH_CYC to 0x1 to generate a 41.6 ns sync time. Example 2, if f_{RCLK} is 6 MHz (period is 166.7 ns), you can configure LATCH_CYC to 0x0 to generate a 166.7 ns sync time. See the "Clocking" chapter or chip-specific DAC information for more on the RCLK frequency. **NOTE** 12-bit DAC adds (LATCH_CYC+1) RCLK cycles before the analog starts to set up the output. Therefore, when you write to this field, NXP recommends using the smallest possible value that still meets the requirement of 40 ns sync time.
7 —	Reserved
6 PTGEN	DAC Periodic Trigger Mode Enable Enables Periodic Trigger Mode when FIFOEN becomes 1. 0b - Disables 1b - Enables
5 TRGSEL	DAC Trigger Select Selects the trigger source for FIFO mode. 0b - Hardware trigger 1b - Software trigger
4 SWMD	Swing Back Mode Specifies whether Swing Back mode is enabled. Selects the Swing Back mode when FIFOEN becomes 1. 0b - Disables 1b - Enables
3 FIFOEN	FIFO Enable Selects the FIFO mode or Buffer mode. 0b - Enables FIFO mode and disables Buffer mode. Any data written to DATA[DATA] goes to buffer then goes to conversion. 1b - Enables FIFO mode. Data will be first read from FIFO to buffer and then goes to conversion.
2-1 DACRFS	DAC Reference Select Selects the source of reference voltage high.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Selects VREFH0 as the reference voltage. 01b - Selects VREFH1 as the reference voltage. 10b - Selects VREFH2 as the reference voltage. 11b - Reserved.
0 DACEN	DAC Enable Enables the DAC system. Starts the programmable reference generator operation. **NOTE** For an 8-bit or 16-bit register write, enable this field after configuring other bytes of GCR. You must configure Periodic Trigger Control (PCR) and DAC FIFO Control (FCR) before enabling this field. 0b - Disables 1b - Enables

42.7.1.6 DAC FIFO Control (FCR)

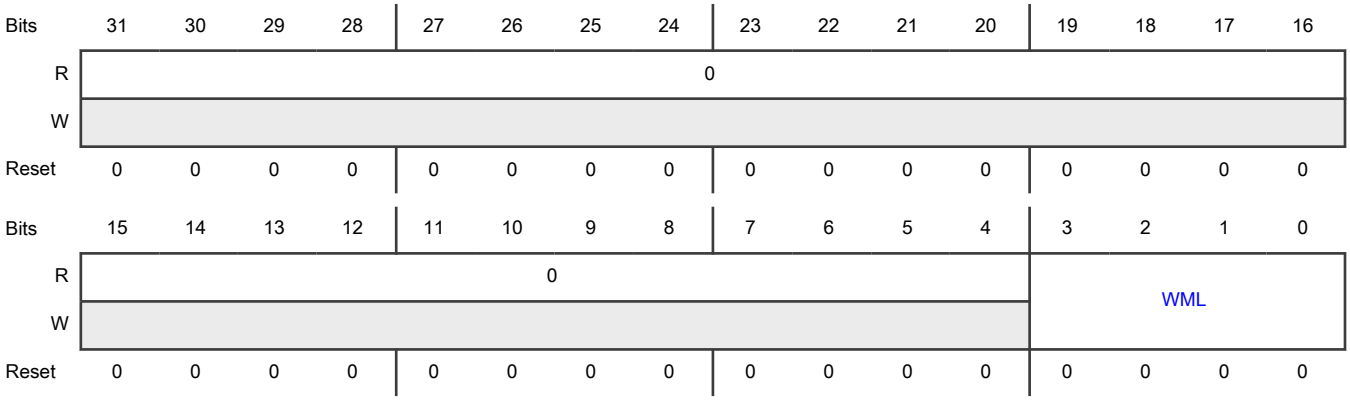
Offset

Register	Offset
FCR	10h

Function

Contains watermark level control for the FIFO.

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 WML	Watermark Level Specifies that when the number of data that remains in FIFO is equal to or less than this level, WM becomes 1. **NOTE** To enable the watermark feature, do not write 0 to this field.

42.7.1.7 DAC FIFO Pointer (FPR)

Offset

Register	Offset
FPR	14h

Function
Contains the read pointer and write pointer of the FIFO.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												FIFO_WPT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												FIFO_RPT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-16	FIFO Write Pointer

Table continues on the next page...

Table continued from the previous page...

Field	Function
FIFO_WPT	Shows the current write pointer of the FIFO.
15-4 —	Reserved
3-0 FIFO_RPT	FIFO Read Pointer Shows the current read pointer of the FIFO.

42.7.1.8 FIFO Status (FSR)

Offset

Register	Offset
FSR	18h

Function

Contains the status flags of the FIFO.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							PTGCOCO	UF	OF	0		SWBK	WM	EMPTY	FULL
W								W1C	W1C	W1C			W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-9 —	Reserved
8 PTGCOCO	Period Trigger Mode Conversion Complete Flag Indicates that PTG mode conversion is completed. This flag becomes 0 by writing a 1 to it.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not completed or not started 1b - Completed
7 UF	FIFO Underflow Flag Specifies that there is a new trigger after EMPTY becomes 1. The FIFO read pointer do not increase in this case and the data sent to DAC analog conversion will not changed. This flag becomes 0 by writing a 1 to it. 0b - No underflow has occurred since the last time the flag was cleared 1b - At least one trigger underflow has occurred since the last time the flag was cleared
6 OF	FIFO Overflow Flag Indicates that data is intended to write into FIFO after FULL becomes 1. The writer pointer do not increase in this case. Do not write the extra data into the FIFO. This flag clears by writing a 1 to it. 0b - No overflow has occurred since the last time the flag was cleared 1b - At least one FIFO overflow has occurred since the last time the flag was cleared
5-4 —	Reserved
3 SWBK	Swing Back One Cycle Complete Flag Indicates that the DAC has completed one period of conversion in Swing Back mode. The read pointer increases to the top (write pointer) once and then decreases to zero. For example, after writing three data to FIFO, the writer pointer is now 3. Then, if continually triggered, the read pointer will swing like: 0-1-2-1-0-1-2-, and so on. After the fourth trigger, the flag sets. Write a 1 to clear this flag. 0b - No swing back cycle has completed since the last time the flag was cleared 1b - At least one swing back cycle has occurred since the last time the flag was cleared
2 WM	FIFO Watermark Status Flag Write 1 to this field if the remaining data in FIFO is less than or equal to WML. By writing data into FIFO by DMA or CPU, this flag clears automatically when the data in FIFO is more than WML. 0b - Data in FIFO is more than watermark level 1b - Data in FIFO is less than or equal to watermark level
1 EMPTY	FIFO Empty Flag Indicates whether the FIFO is empty or not. 12-bit DAC automatically clears this flag when CPU or DMA writes data to the FIFO. 0b - Not empty 1b - Empty
0	FIFO Full Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
FULL	Indicates whether the FIFO is full or not. FIFO read with software trigger or hardware trigger automatically clears this flag. 0b - Not full 1b - Full

42.7.1.9 Interrupt Enable (IER)

Offset

Register	Offset
IER	1Ch

Function

Contains the interrupt enable fields.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							PTGCOCO...	UF_IE	OF_IE	0		SWBK_IE	WM_IE	EMPTY_IE	FULL_IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-9 —	Reserved
8 PTGCOCO_IE	PTG Mode Conversion Complete Interrupt Enable Enables interrupt after the PTG mode conversion completes. 0b - Disables 1b - Enables

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 UF_IE	FIFO Underflow Interrupt Enable Enables interrupt when FIFO underflows. 0b - Disables 1b - Enables
6 OF_IE	FIFO Overflow Interrupt Enable Enables interrupt when FIFO overflows. 0b - Disables 1b - Enables
5-4 —	Reserved
3 SWBK_IE	Swing Back One Cycle Complete Interrupt Enable Enables interrupt after the swing back one cycle completes. 0b - Disables 1b - Enables
2 WM_IE	FIFO Watermark Interrupt Enable Enables interrupt when data in FIFO is less than or equal to watermark level. 0b - Disables 1b - Enables
1 EMPTY_IE	FIFO Empty Interrupt Enable Enables interrupt when FIFO is empty. 0b - Disables 1b - Enables
0 FULL_IE	FIFO Full Interrupt Enable Enables interrupt when FIFO is full. 0b - Disables 1b - Enables

42.7.1.10 DMA Enable (DER)

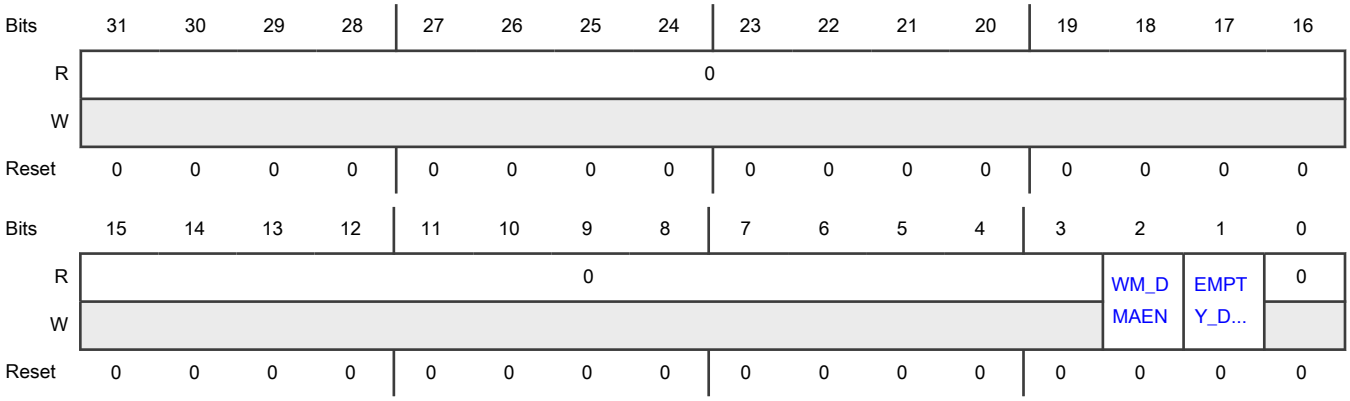
Offset

Register	Offset
DER	20h

Function

Contains the DMA enable fields.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 WM_DMAEN	FIFO Watermark DMA Enable Enables DMA request when data in FIFO is less than or equal to watermark level. 0b - Disables 1b - Enables
1 EMPTY_DMAEN	FIFO Empty DMA Enable Enables DMA request when FIFO is empty. 0b - Disables 1b - Enables
0 —	Reserved

42.7.1.11 Reset Control (RCR)

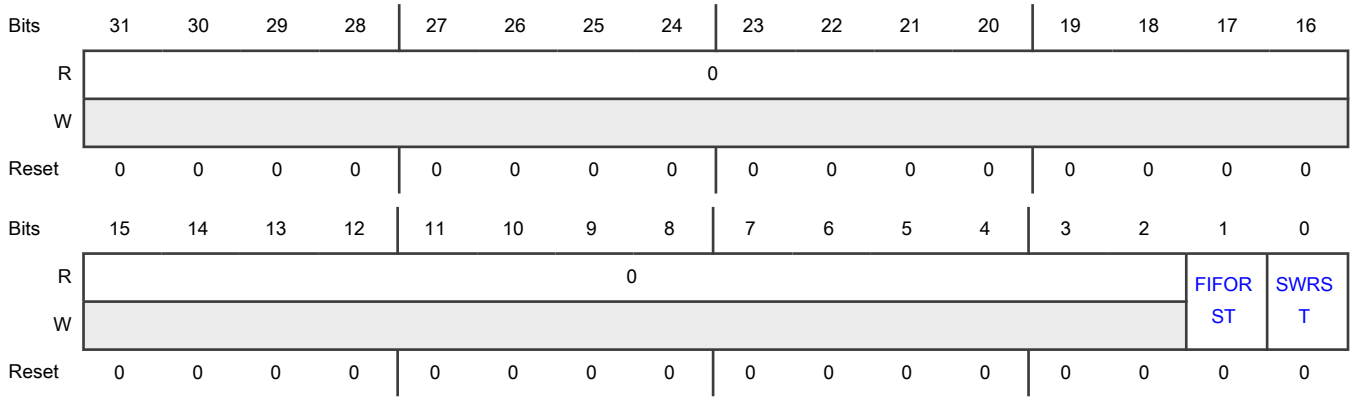
Offset

Register	Offset
RCR	24h

Function

Contains the software reset and FIFO reset fields.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 FIFORST	FIFO Reset Resets the FIFO pointers and flags in FIFO Status (FSR). Remains 1 until you write 0 to it. 0b - No effect 1b - FIFO reset
0 SWRST	Software Reset Resets all DAC registers and internal logic. Remains 1 until you write 0 to it. 0b - No effect 1b - Software reset

42.7.1.12 Trigger Control (TCR)

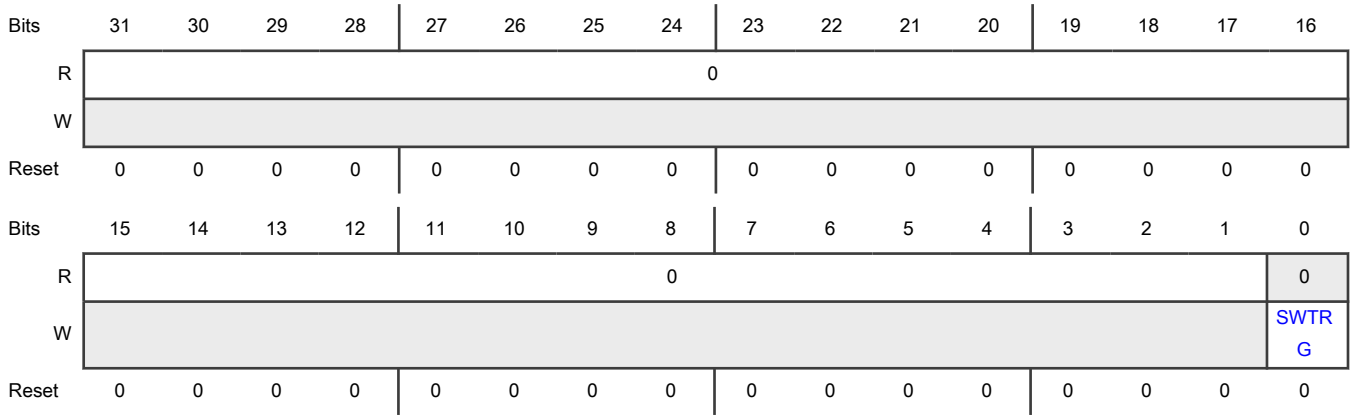
Offset

Register	Offset
TCR	28h

Function

Contains the software trigger field.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 SWTRG	Software Trigger Indicates whether the DAC soft trigger is valid. Active high. This is a write-only field, which always reads 0. If TRGSEL is 1 and FIFO is enabled and not empty, writing 1 to this field advances the FIFO read pointer once. **NOTE** During continuous conversions, do not adopt hardware trigger and software trigger in a mixed use. 0b - Not valid 1b - Valid

42.7.1.13 Periodic Trigger Control (PCR)

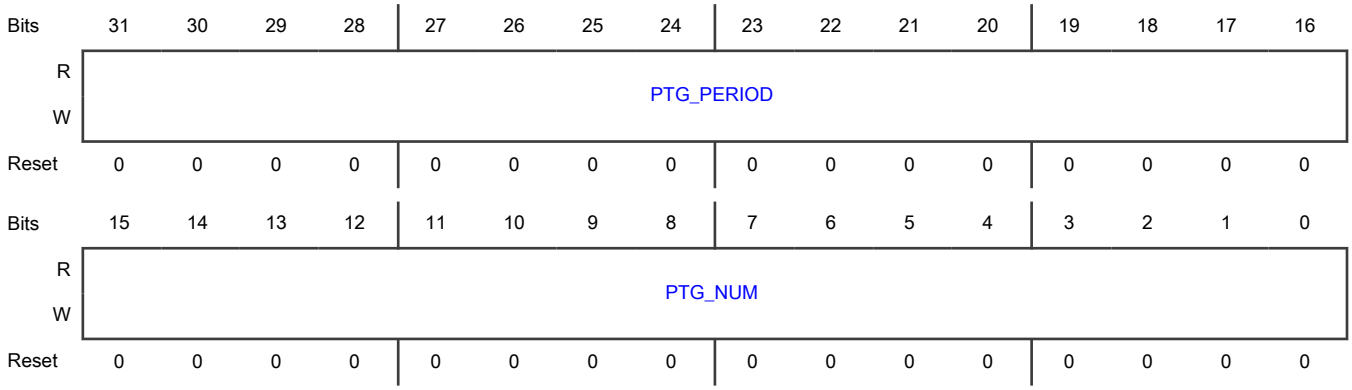
Offset

Register	Offset
PCR	2Ch

Function

Contains the periodic trigger period and periodic trigger number fields.

Diagram



Fields

Field	Function
31-16 PTG_PERIOD	Periodic Trigger Period Width Uses only in Periodic Trigger mode. It controls the periodic trigger frequency. There are [PTG_PERIOD+1] RCLK cycles between each periodic trigger. See the chip-specific DAC information for more on the RCLK frequency. Example 1, if f_{RCLK} is 48 MHz and the conversion speed is 0.5 MHz, there should be 96 RCLK cycles between each periodic trigger. So you must configure the PTG_PERIOD as 0x5F(95). Example 2, if f_{RCLK} is 6 MHz and the conversion speed is 0.5 MHz, there should be 12 RCLK cycles between each periodic trigger. So you must configure the PTG_PERIOD as 0xB(11). **NOTE** The PTG_PERIOD should not be less than 0x1. **NOTE** The periodic trigger frequency should not be larger than the analog conversion speed.
15-0 PTG_NUM	Periodic Trigger Number Uses only in Periodic Trigger mode. There are [PTG_NUM] internal triggers following the first hardware/software trigger. So, there are [PTG_NUM+1] conversions in total. **NOTE** If you write 0 to it, there will be infinite triggers following the first hardware/software trigger, until PTGEN becomes 0.

Chapter 43
14-bit Digital-to-Analog Converter (DAC)

43.1 Chip-specific 14-bit DAC information

Table 372. Reference links to related information

Topic	Related module	Reference
Full description	14-bit DAC	14-bit Digital-to-Analog Converter (DAC)
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See DAC2_TRIG registers in INPUTMUX

43.1.1 Module instances

This device has one instance of the 14-bit 5 Msps DAC module, DAC2, with compatible DAC Controller. Set controller FIFO size to 5 (32 words).

43.1.2 14-bit DAC references

VIN = VDD_ANA

43.1.3 14-bit DAC chip-specific initialization

To initialize the 14-bit DAC (DAC2) module:

1. Set the clock divider for DAC2 with DACxCLKDIV[DIV] bit in SYSCON.
2. Enable the clock to DAC2 (AHBCLKCTRLn[DACn] = 1) in SYSCON. This enables the register interface and the peripheral function clock.
3. Power up the DAC2 with ACTIVE_CFG1 and LP_CFG1 registers in SPC. See SOC_CNTRL fields.
4. Reset and release the DAC2 peripheral reset by setting the PRESETCTRLSETn[DACn_RST] bit and then set PRESETCTRLCLRn[DACn_RST] bit.
5. Reset the DAC2 logic by setting and clearing bit RCR[SWRST]. See Reset Control (RCR).
6. Reset the DAC2 FIFO by setting and clearing bit RCR[FIFORST].
7. Enable VREF module SOC_CNTRL[VREF_EN] = 1 in ACTIVE_CFG1 register in SPC module. See SOC_CNTRL fields.
8. If needed, initialize the DAC2 reference source before selecting the DAC2 reference with GCR[DACRFS]. See Global Control (GCR)
   - If using VREF_OUT as reference, see Chip-specific VREF information link to VREF_OUT initialization
9. Configure DAC2 as described in section Initialization

43.1.4 14-bit DAC trigger inputs

14-bit DAC (DAC2) trigger sources get routed through the INPUTMUX. See the DAC2_TRIG register in the INPUTMUX chapter for available trigger sources.

43.2 Overview

The 14-bit general-purpose digital-to-analog converter is a High-performance DAC. The output of the DAC can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.

43.2.1 Block diagram

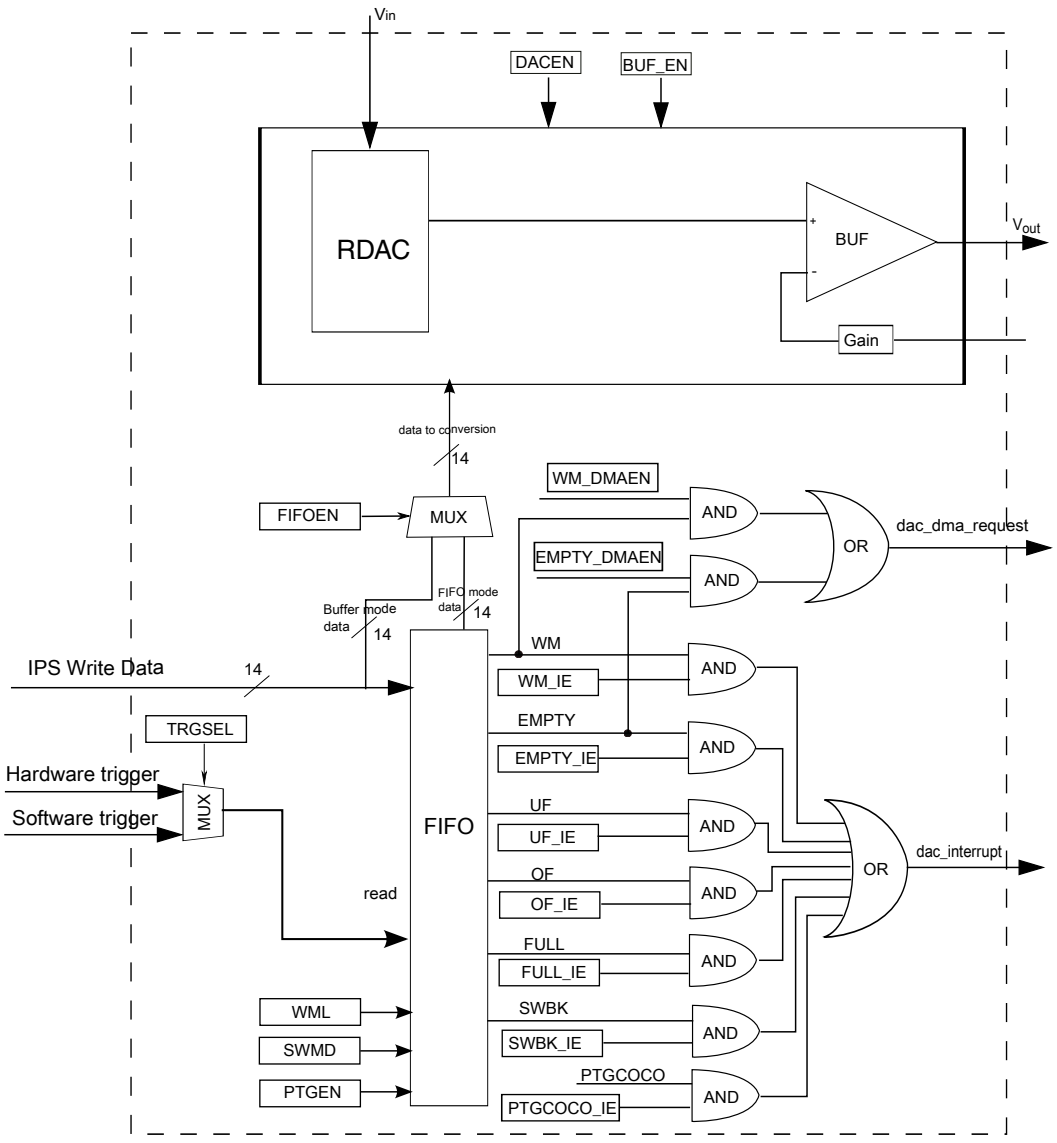


Figure 187. 14-bit DAC block diagram

43.2.2 Features

- Includes an on-chip programmable reference generator output. The voltage output ranges from 1/16384 V_{in} to V_{in}, and the step is 1/16384 V_{in}, where V_{in} is the input reference voltage.
- Supports 32-word depth FIFO with the configurable watermark.
- Uses multiple operation modes.

- Buffer mode
- FIFO mode
- Swing Back mode
- Periodic Trigger mode

- Supports software trigger and hardware trigger.
- Provides interrupt and DMA support.

43.3 Functional description

14-bit DAC converts a 14-bit data to a stepped analog output voltage when enabled. The output voltage range is from V_{in} to $V_{in}/16384$, and the step is $V_{in}/16384$.

You can configure the 14-bit DAC to Buffer mode, FIFO mode, Swing Back mode, or Periodic Trigger mode.

NOTE

You must configure 14-bit DAC before any conversion. During conversion or FIFO write operation, you must not change the configuration or disable DACEN. If you switch to another mode, you must first write a 1 to SWRST to reset 14-bit DAC or write 0 to DACEN.

43.3.1 Buffer mode

You can enable Buffer mode with DACEN.

In Buffer mode, any write to DATA pushes the data to the analog conversion without trigger support.

NOTE

The data write speed must not be higher than the analog conversion speed.

43.3.2 FIFO mode

You can enable FIFO mode with FIFOEN and DACEN.

In FIFO mode, for a 32-/16-bit write to DATA, put the data into the FIFO and the write pointer increments automatically. For an 8-bit write, only if you write both the bytes, the write pointer increases. You must ensure that an 8-bit write is done in pair and write both the upper and lower bytes. There is no priority, which byte to write first.

When a software or hardware trigger comes, the read pointer pointed data in FIFO pushes to the analog conversion, and the read pointer increments automatically. If the trigger comes when FIFO is empty, the data to analog remains unchanged, and an underflow error flag sets.

43.3.3 Swing Back mode

You can enable Swing Back mode with FIFOEN, SWMD, and DACEN.

The read pointer swings between the writer pointer and zero in Swing Back mode. The trigger increases the read pointer till the writer pointer and decreases the read pointer till zero, and so on. EMPTY, FULL, and WM have no meaning and you should ignore them during Swing Back mode.

NOTE

In Swing Back mode, the FIFO should first write at least two data; otherwise, this mode will not work correctly. You should not write the FIFO when Swing Back mode is in progress.

43.3.4 Periodic Trigger mode

You can enable Periodic Trigger mode with FIFOEN, PTGEN, and DACEN.

Due to the trigger (hardware or software) synchronization logic, there is a delay time variation between each conversion. If there is a need to trigger DAC periodically, NXP recommends you to use Periodic Trigger mode.

In Periodic Trigger mode, you only send the first trigger. Then after every [PTG_PERIOD+1] read clock (RCLK) cycle, an internal trigger automatically triggers the DAC. There are [PTG_NUM] internal triggers, therefore, in total [PTG_NUM+1] conversions, including the first trigger that you send. Write 0 to PTGEN to terminate the current conversion queue. Then, after the current conversion completes, the conversion terminates, and the PTGCOCO sets. If PTG_NUM becomes zero, the infinite triggers follow the first hardware/software trigger until you write 0 to PTGEN. In any case, FIFORST or SWRST can terminate the conversion.

During the periodic trigger mode conversion, you can ignore the additional hardware/software triggers. You can start another queue after PTGCOCO becomes 1.

During Periodic Trigger Mode, WM feeds data to FIFO to avoid underflow.

You can combine Periodic Trigger mode with SWMD to periodically trigger DAC and swing back.

43.3.5 Low-power mode operation

If the RCLK remains enabled, 14-bit DAC can be functional with FIFO mode or Swing Back mode or Periodic Trigger mode during Stop and Wait modes, using the hardware trigger initiating the conversion.

43.3.6 DMA

After the corresponding DMA enable register field becomes 1, you can automatically generate DMA requests when WM or EMPTY becomes 1. When you enable DMA, the error flags can only generate an interrupt.

43.3.7 Clocks

14-bit DAC block requires the following clock(s):

- Bus clock for register access and writing data to FIFO or Buffer.
- Read clock (RCLK) to push data from FIFO to the analog DAC.

43.3.8 Resets

The following are the 14-bit DAC resets:

1. Global reset fed into the block. This global reset resets everything.
2. SWRST: resets all 14-bit DAC registers and internal logic.
3. FIFORST: resets the FIFO pointers in DAC FIFO Pointer (FPR) and flags in FIFO Status (FSR).

43.3.9 Interrupts

After the corresponding interrupt enable register field becomes 1, warning flags and error flags can generate interrupt. The warning flags are EMPTY, FULL, WM, SWBK, and PTGCOCO. The error flags are OF and UF.

43.4 DAC external signal descriptions

Table 373. DAC external signal descriptions

Signal	Description	I/O
Vin	DAC reference high voltage	I
Vout	DAC output	O

43.5 Initialization

To initialize 14-bit DAC:

1. Configure the read clock at chip level.
2. Configure Work mode and GCR enables 14-bit DAC.
3. Configure Interrupt Enable (IER) if required.
4. Configure DMA Enable (DER) if required.
5. Configure Periodic Trigger Control (PCR) if required.
6. Write data to DATA to fill the FIFO, if FIFO mode.
7. Wait analog start-up time
8. Use hardware or software trigger to push data to the analog DAC to convert, if FIFO mode.
9. Write data directly to DATA to push data to the analog DAC to convert, if Buffer mode.

43.6 Application Information

Digital delay

Due to the resynchronization, there will be a delay between the trigger posedge and analog voltage settling. The delay is 2T to 3T, where T is the RCLK period. So the delay has one T uncertainty. It is obvious that fast RCLK can achieve small delay.

RCLK frequency

For discrete conversion, the RCLK frequency has no low limit. But slow RCLK involves large digital delay and uncertainty.

For continuous conversion, the RCLK frequency should be at least 5 times the trigger speed.

For continuous conversion, due to the one cycle uncertainty RCLK, the trigger period should not be less than (200ns + T) where 200ns period means the typical analog conversion speed 5 Msps.

To achieve the full speed of DAC analog, you must configure to the internal periodic trigger mode.

For Internal Periodic Trigger mode, RCLK frequency should be at least 1 times the conversion speed. For example, to achieve 5 Msps conversion speed, at least 5 MHz RCLK is required in periodic trigger mode.

43.7 Memory map and register definition

All registers are accessible in 8-, 16-, or 32-bit accesses. For efficiency, 32-bit accesses are recommended. Access to an address outside the valid memory map generates a bus error.

43.7.1 14-bit DAC register descriptions

Access to an address outside the valid memory map generates a bus error.

43.7.1.1 DAC memory map

DAC2 base address: 4011_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version Identifier (VERID)	32	R	0100_0000h
4h	Parameter (PARAM)	32	R	0000_0004h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
8h	Data (DATA)	32	RW	0000_0000h
Ch	Global Control (GCR)	32	RW	0000_0000h
10h	DAC FIFO Control (FCR)	32	RW	0000_0000h
14h	DAC FIFO Pointer (FPR)	32	R	0000_0000h
18h	FIFO Status (FSR)	32	RW	0000_0002h
1Ch	Interrupt Enable (IER)	32	RW	0000_0000h
20h	DMA Enable (DER)	32	RW	0000_0000h
24h	Reset Control (RCR)	32	RW	0000_0000h
28h	Trigger Control (TCR)	32	RW	0000_0000h
2Ch	Periodic Trigger Control (PCR)	32	RW	0000_0000h

43.7.1.2 Version Identifier (VERID)

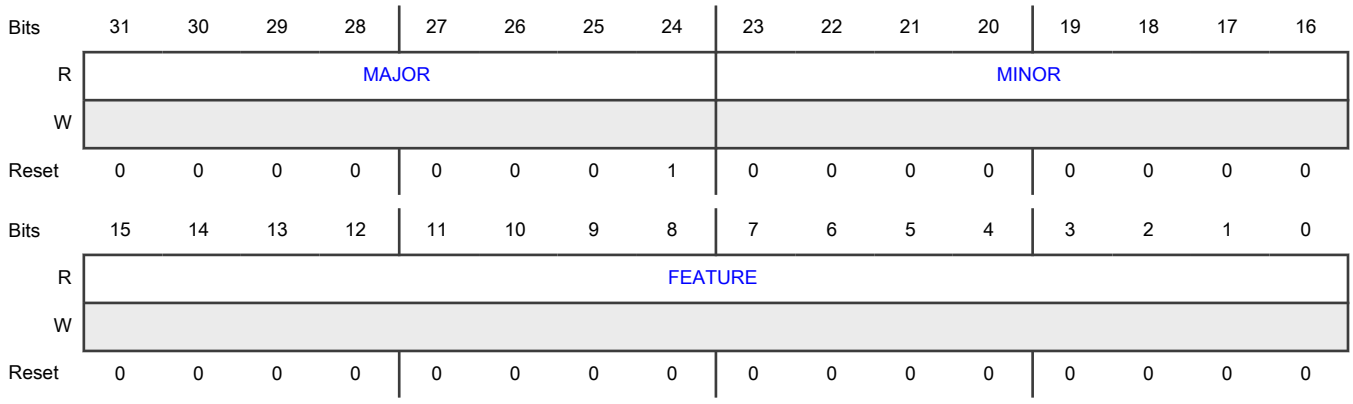
Offset

Register	Offset
VERID	0h

Function

Indicates the version integrated for this instance on the chip.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15-0 FEATURE	Feature Identification Number Returns the feature set number.

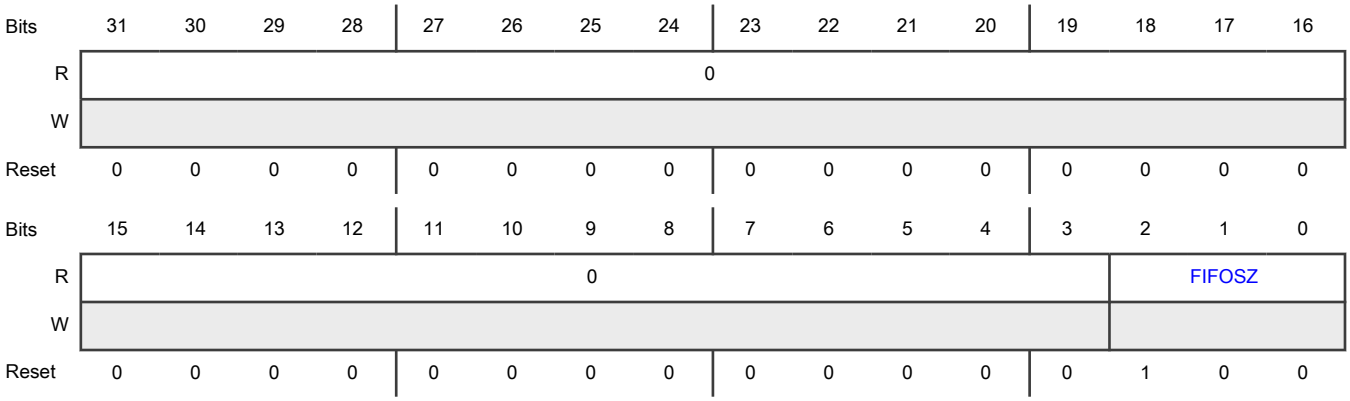
43.7.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function
Indicates the feature parameters for this instance on the chip.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 FIFOSZ	FIFO Size Indicates that the FIFO depth is 2^(FIFOSZ+1) words.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000b - Reserved 001b - FIFO depth is 4 010b - FIFO depth is 8 011b - FIFO depth is 16 100b - FIFO depth is 32 101b - FIFO depth is 64 110b - FIFO depth is 128 111b - FIFO depth is 256

43.7.1.4 Data (DATA)

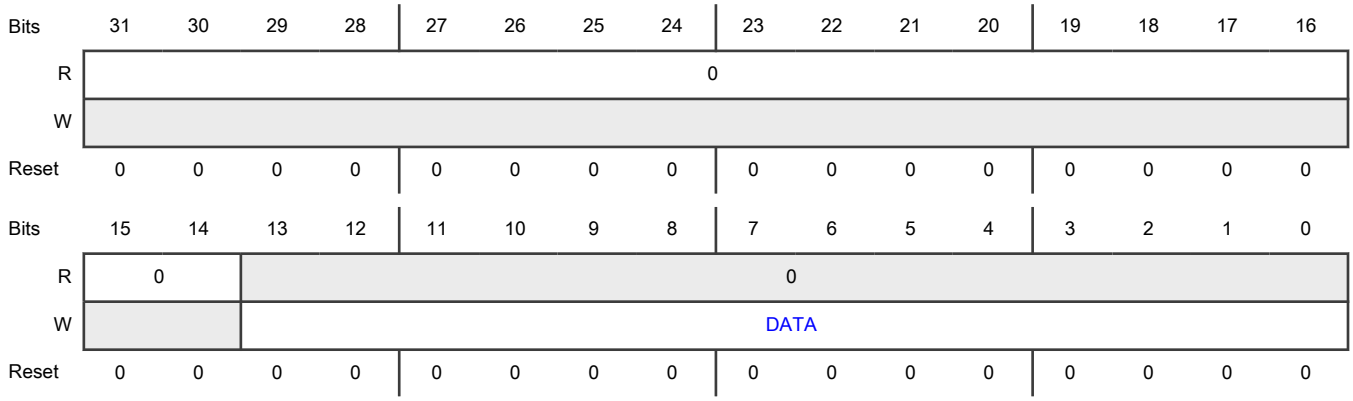
Offset

Register	Offset
DATA	8h

Function

Holds the FIFO entry in FIFO mode and serves as the buffer in Buffer mode.

Diagram



Fields

Field	Function
31-14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-0 DATA	FIFO Entry or Buffer Entry Specifies that in FIFO mode, Swing Back mode, or Periodic Trigger mode, this is the FIFO data entry. In Buffer mode, write to this field pushes the data to analog without trigger support.

43.7.1.5 Global Control (GCR)

Offset

Register	Offset
GCR	Ch

Function

Contains configuration options for DAC function, such as mode select, trigger select, and so on.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0														BUF_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0	PTGEN	TRGSEL	SWMD	FIFOEN	0		DACEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-18 —	Reserved
17 BUF_EN	Buffer Enable Uses OPAMP as the DAC analog buffer. 0b - Not used 1b - Used
16-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7 —	Reserved
6 PTGEN	DAC Periodic Trigger Mode Enable Enables Periodic Trigger Mode when FIFOEN becomes 1. 0b - Disables 1b - Enables
5 TRGSEL	DAC Trigger Select Selects the trigger source for FIFO mode. 0b - Hardware trigger 1b - Software trigger
4 SWMD	Swing Back Mode Specifies whether Swing Back mode is enabled. Selects the Swing Back mode when FIFOEN becomes 1. 0b - Disables 1b - Enables
3 FIFOEN	FIFO Enable Selects the FIFO mode or Buffer mode. 0b - Enables FIFO mode and disables Buffer mode. Any data written to DATA[DATA] goes to buffer then goes to conversion. 1b - Enables FIFO mode. Data will be first read from FIFO to buffer and then goes to conversion.
2-1 —	Reserved
0 DACEN	DAC Enable Enables the DAC system. Starts the programmable reference generator operation. **NOTE** For an 8-bit or 16-bit register write, enable this field after configuring other bytes of GCR. You must configure Periodic Trigger Control (PCR) and DAC FIFO Control (FCR) before enabling this field. 0b - Disables 1b - Enables

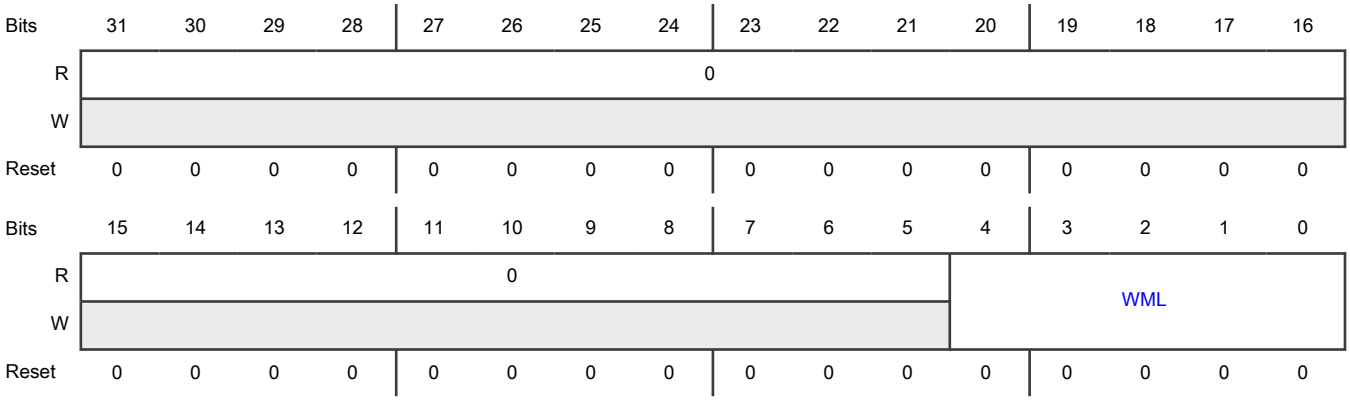
43.7.1.6 DAC FIFO Control (FCR)

Offset

Register	Offset
FCR	10h

Function
Contains watermark level control for the FIFO.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 WML	Watermark Level Specifies that when the number of data that remains in FIFO is equal to or less than this level, WM becomes 1. **NOTE** To enable the watermark feature, do not write 0 to this field.

43.7.1.7 DAC FIFO Pointer (FPR)

Offset

Register	Offset
FPR	14h

Function
Contains the read pointer and write pointer of the FIFO.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												FIFO_WPT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												FIFO_RPT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved
20-16 FIFO_WPT	FIFO Write Pointer Shows the current write pointer of the FIFO.
15-5 —	Reserved
4-0 FIFO_RPT	FIFO Read Pointer Shows the current read pointer of the FIFO.

43.7.1.8 FIFO Status (FSR)

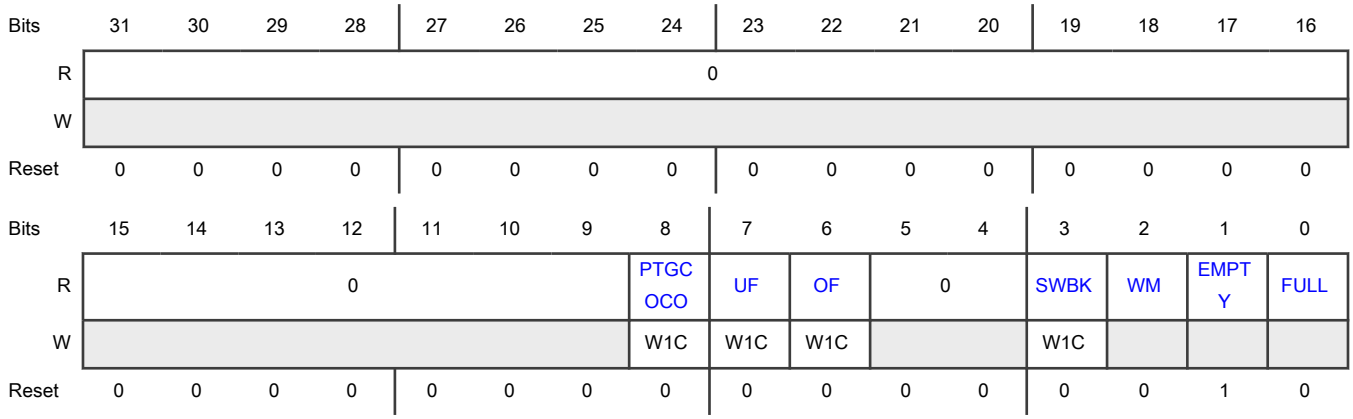
Offset

Register	Offset
FSR	18h

Function

Contains the status flags of the FIFO.

Diagram



Fields

Field	Function
31-9 —	Reserved
8 PTGCOCO	Period Trigger Mode Conversion Complete Flag Indicates that PTG mode conversion is completed. This flag becomes 0 by writing a 1 to it. 0b - Not completed or not started 1b - Completed
7 UF	FIFO Underflow Flag Specifies that there is a new trigger after EMPTY becomes 1. The FIFO read pointer do not increase in this case and the data sent to DAC analog conversion will not changed. This flag becomes 0 by writing a 1 to it. 0b - No underflow has occurred since the last time the flag was cleared 1b - At least one trigger underflow has occurred since the last time the flag was cleared
6 OF	FIFO Overflow Flag Indicates that data is intended to write into FIFO after FULL becomes 1. The writer pointer do not increase in this case. Do not write the extra data into the FIFO. This flag clears by writing a 1 to it. 0b - No overflow has occurred since the last time the flag was cleared 1b - At least one FIFO overflow has occurred since the last time the flag was cleared
5-4 —	Reserved
3 SWBK	Swing Back One Cycle Complete Flag Indicates that the DAC has completed one period of conversion in Swing Back mode. The read pointer increases to the top (write pointer) once and then decreases to zero. For example, after writing three

Table continues on the next page...

Table continued from the previous page...

Field	Function
	data to FIFO, the writer pointer is now 3. Then, if continually triggered, the read pointer will swing like: 0-1-2-1-0-1-2-, and so on. After the fourth trigger, the flag sets. Write a 1 to clear this flag. 0b - No swing back cycle has completed since the last time the flag was cleared 1b - At least one swing back cycle has occurred since the last time the flag was cleared
2 WM	FIFO Watermark Status Flag Write 1 to this field if the remaining data in FIFO is less than or equal to WML. By writing data into FIFO by DMA or CPU, this flag clears automatically when the data in FIFO is more than WML. 0b - Data in FIFO is more than watermark level 1b - Data in FIFO is less than or equal to watermark level
1 EMPTY	FIFO Empty Flag Indicates whether the FIFO is empty or not. 14-bit DAC automatically clears this flag when CPU or DMA writes data to the FIFO. 0b - Not empty 1b - Empty
0 FULL	FIFO Full Flag Indicates whether the FIFO is full or not. FIFO read with software trigger or hardware trigger automatically clears this flag. 0b - Not full 1b - Full

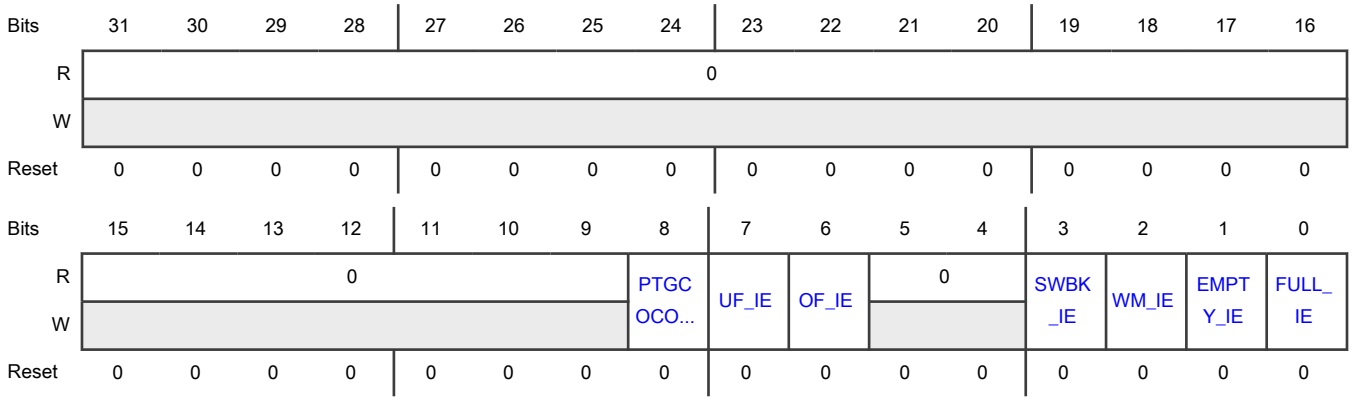
43.7.1.9 Interrupt Enable (IER)

Offset

Register	Offset
IER	1Ch

Function
Contains the interrupt enable fields.

Diagram



Fields

Field	Function
31-9 —	Reserved
8 PTGCOCO_IE	PTG Mode Conversion Complete Interrupt Enable Enables interrupt after the PTG mode conversion completes. 0b - Disables 1b - Enables
7 UF_IE	FIFO Underflow Interrupt Enable Enables interrupt when FIFO underflows. 0b - Disables 1b - Enables
6 OF_IE	FIFO Overflow Interrupt Enable Enables interrupt when FIFO overflows. 0b - Disables 1b - Enables
5-4 —	Reserved
3 SWBK_IE	Swing Back One Cycle Complete Interrupt Enable Enables interrupt after the swing back one cycle completes. 0b - Disables 1b - Enables
2	FIFO Watermark Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
WM_IE	Enables interrupt when data in FIFO is less than or equal to watermark level. 0b - Disables 1b - Enables
1 EMPTY_IE	FIFO Empty Interrupt Enable Enables interrupt when FIFO is empty. 0b - Disables 1b - Enables
0 FULL_IE	FIFO Full Interrupt Enable Enables interrupt when FIFO is full. 0b - Disables 1b - Enables

43.7.1.10 DMA Enable (DER)

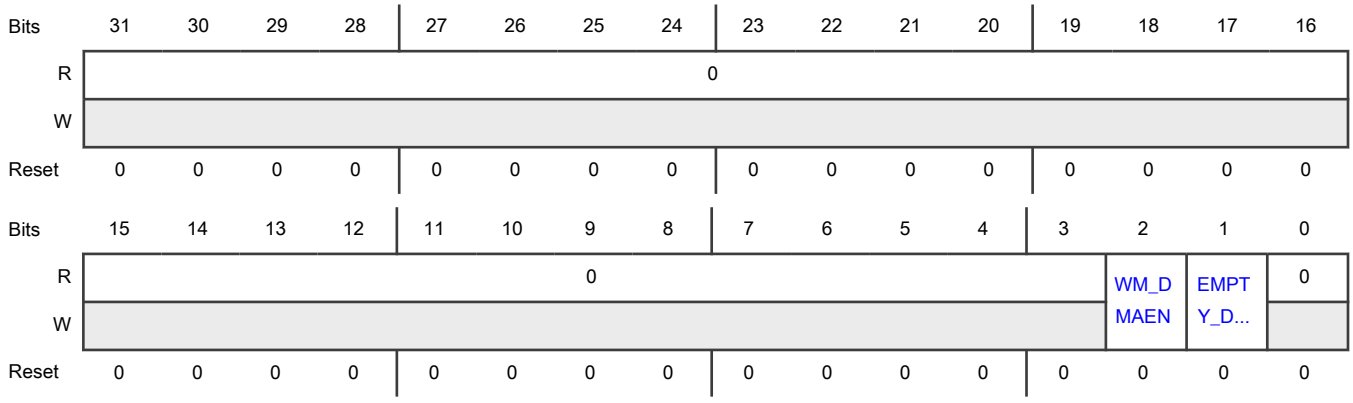
Offset

Register	Offset
DER	20h

Function

Contains the DMA enable fields.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 WM_DMAEN	FIFO Watermark DMA Enable Enables DMA request when data in FIFO is less than or equal to watermark level. 0b - Disables 1b - Enables
1 EMPTY_DMAEN	FIFO Empty DMA Enable Enables DMA request when FIFO is empty. 0b - Disables 1b - Enables
0 —	Reserved

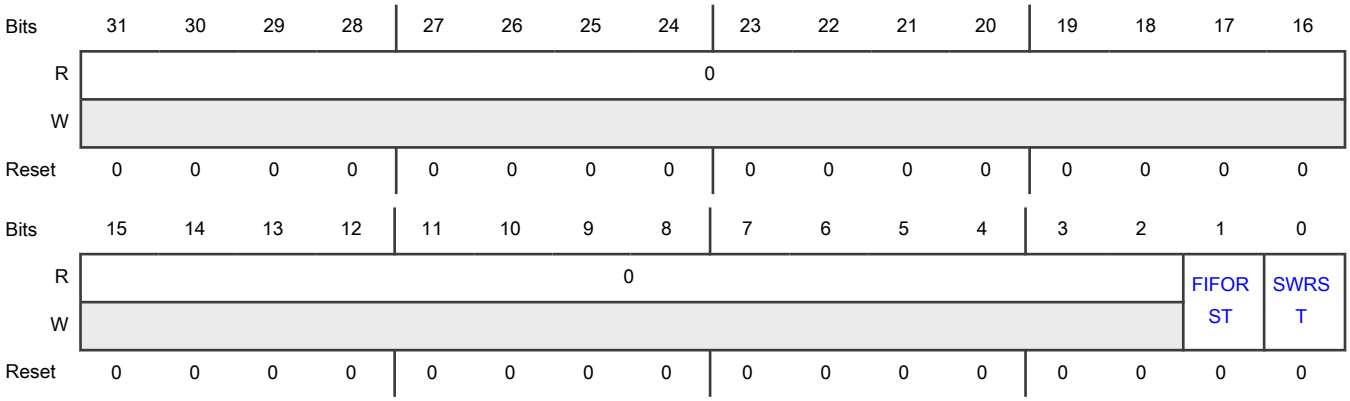
43.7.1.11 Reset Control (RCR)

Offset

Register	Offset
RCR	24h

Function
Contains the software reset and FIFO reset fields.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 FIFORST	FIFO Reset Resets the FIFO pointers and flags in FIFO Status (FSR). Remains 1 until you write 0 to it. 0b - No effect 1b - FIFO reset
0 SWRST	Software Reset Resets all DAC registers and internal logic. Remains 1 until you write 0 to it. 0b - No effect 1b - Software reset

43.7.1.12 Trigger Control (TCR)

Offset

Register	Offset
TCR	28h

Function
Contains the software trigger field.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															0
W																SWTRG
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 SWTRG	Software Trigger Indicates whether the DAC soft trigger is valid. Active high. This is a write-only field, which always reads 0. If TRGSEL is 1 and FIFO is enabled and not empty, writing 1 to this field advances the FIFO read pointer once. **NOTE** During continuous conversions, do not adopt hardware trigger and software trigger in a mixed use. 0b - Not valid 1b - Valid

43.7.1.13 Periodic Trigger Control (PCR)

Offset

Register	Offset
PCR	2Ch

Function
Contains the periodic trigger period and periodic trigger number fields.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PTG_PERIOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PTG_NUM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 PTG_PERIOD	Periodic Trigger Period Width Uses only in Periodic Trigger mode. It controls the periodic trigger frequency. There are [PTG_PERIOD+1] RCLK cycles between each periodic trigger. See the chip-specific DAC information for more on the RCLK frequency. Example 1, if f_{RCLK} is 48 MHz and the conversion speed is 0.5 MHz, there should be 96 RCLK cycles between each periodic trigger. So you must configure the PTG_PERIOD as 0x5F(95). Example 2, if f_{RCLK} is 6 MHz and the conversion speed is 0.5 MHz, there should be 12 RCLK cycles between each periodic trigger. So you must configure the PTG_PERIOD as 0xB(11). **NOTE** The periodic trigger frequency should not be larger than the analog conversion speed.
15-0 PTG_NUM	Periodic Trigger Number Uses only in Periodic Trigger mode. There are [PTG_NUM] internal triggers following the first hardware/software trigger. So, there are [PTG_NUM+1] conversions in total. **NOTE** If you write 0 to it, there will be infinite triggers following the first hardware/software trigger, until PTGEN becomes 0.

Chapter 44
Low-Power Comparator (CMP)

44.1 Chip-specific CMP information

Table 374. Reference links to related information

Topic	Related module	Reference
Full description	CMP	CMP
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See CMPn_TRIG registers in INPUTMUX

NOTE

See AHB Clock Control 3 (AHBCLKCTRL3) to enable CMP2 clock. Also see Peripheral Reset Control 3 (PRESETCTRL3) for reset control.

44.1.1 Module instances

This chip supports three instances of the CMP module, CMP0, CMP1, and CMP2. Each comparator has an 8-bit DAC.

NOTE

CMP can be functional in low power (LP) mode (including Deep Sleep, Power Down or Deep Power Down mode). To enable CMP in the LP mode, you should enable the CMPn_EN bitfield in the LP_CFG1 register of SPC. When chip enters the LP mode, you must ensure the clock source of the required function is active.

44.1.2 CMP input connections

Table 375. CMP input connections

Index	CMP0_Positive	CMP0_Negative	CMP1_Positive	CMP1_Negative	CMP2_Positive	CMP2_Negative	Description
0	CMP0_IN0P (P1_0)	CMP0_IN0N (P1_0)	CMP1_IN0P (P1_1)	CMP1_IN0N (P1_1)	CMP2_IN0P (P1_2)	CMP2_IN0N (P1_2)	VDD domain
1	CMP0_IN1P (P1_3)	CMP0_IN1N (P1_3)	CMP1_IN1P (P0_3)	CMP1_IN1N (P0_3)	CMP2_IN1P (P0_7)	CMP2_IN1N (P0_7)	VDD domain
2	CMP0_IN2P (P1_4)	CMP0_IN2N (P1_4)	CMP1_IN2P (P0_22)	CMP1_IN2N (P0_22)	CMP2_IN2P (P0_23)	CMP2_IN2N P0_23)	VDD domain
3	CMP0_IN3P (P1_5)	CMP0_IN3N (P1_5)	CMP1_IN3P (P1_20)	CMP1_IN3N (P1_20)	CMP2_IN3P (P1_21)	CMP2_IN3N (P1_21)	VDD domain

Table continues on the next page...

Table 375. CMP input connections (continued)

Index	CMP0_Positive	CMP0_Negative	CMP1_Positive	CMP1_Negative	CMP2_Positive	CMP2_Negative	Description
4	CMP0_IN4P/ OPAMP0_OUT (P4_15)	CMP0_IN4N/ DAC0_OUT (P4_2)	CMP1_IN4P/ OPAMP1_OUT (P4_19)	CMP1_IN4N/ DAC0_OUT (P4_2)	CMP2_IN4P/ OPAMP2_OUT (P4_23)	CMP2_IN4N/ DAC0_OUT (P4_2)	VDD_ANA domain
5	OPAMP0_INT	CMP0_IN5N/ DAC1_OUT (P4_3)	OPAMP1_INT	CMP1_IN5N/ DAC1_OUT (P4_3)	OPAMP2_INT	CMP2_IN5N/ DAC1_OUT (P4_3)	VDD_ANA domain
7	CMP0 DAC	CMP0 DAC	CMP1 DAC	CMP1 DAC	CMP2 DAC	CMP2 DAC	Internal

44.1.3 External trigger sources

RRCR0[RR_EXTTRG_SEL] controls the trigger source for the CMP. Note that only CMP2 supports multiple options. RRCR0[RR_EXTTRG_SEL] is not writeable in CMP0 and CMP1 and only source '0' is available.

Table 376. External trigger sources

CMP instance	Source number	RR_EXTTRG_SEL trigger source
CMP0	0	LPTMR0
CMP1	0	LPTMR1
CMP2	0	SCT_OUT4
	1	SCT_OUT5
	2	SCT_OUT8
	3	CTIMER0_MAT3
	4	CTIMER1_MAT3
	5	CTIMER2_MAT3
	6	CTIMER0_MAT2
	7	CTIMER4_MAT2
	8	PWM0_SM0_MUX_TRIG0/PWM0_SM0_MUX_TRIG1
	9	PWM0_SM1_MUX_TRIG0/PWM0_SM1_MUX_TRIG1
	10	PWM0_SM2_MUX_TRIG0/PWM0_SM2_MUX_TRIG1
	11	PWM0_SM3_MUX_TRIG0/PWM0_SM3_MUX_TRIG1
	12	PWM1_SM0_MUX_TRIG0/PWM1_SM0_MUX_TRIG1
	13	PWM1_SM1_MUX_TRIG0/PWM1_SM1_MUX_TRIG1
	14	PWM1_SM2_MUX_TRIG0/PWM1_SM2_MUX_TRIG1
	15	PWM1_SM3_MUX_TRIG0/PWM1_SM3_MUX_TRIG1

44.1.4 Input references

In the DAC Control Register, when:

- VRSEL=0, VREFH0 is selected, which is VDD
- VRSEL=1, VREFH1 is selected, which is VREFO

44.1.5 Functional clock selection

CCR1[FUNC_CLK_SEL] selects clock sources as follows:

- 00b - Reserved
- 01b - FRO_16K
- 10b - XTAL32K
- 11b - CMPn function clock

44.1.6 Round robin clock selection

RRCR0[RR_CLK_SEL] selects clock sources as follows:

- 00b - Reserved
- 01b - FRO_16K
- 10b - XTAL32K
- 11b - CMPn round robin clock

44.2 Overview

The LPCMP module provides a circuit to compare two analog input voltages. It includes the following:

- A low power comparator (CMP)
- A DAC
- An analog mux (ANMUX)

See Block diagram for more information.

LPCMP can operate across the full range of the supply voltage, known as rail-to-rail operation.

DAC is a 256-tap resistor ladder network that provides a selectable voltage reference for applications requiring a voltage reference. DAC divides the supply reference V_{in} into 256 voltage levels. An 8-bit digital signal input selects the output voltage level, which varies from V_{in} to $V_{in}/256$.

You can select V_{in} from the following voltage sources:

- VREFH0
- VREFH1

See the Chip-specific LPCMP information for more information on source of VREFH0 and VREFH1.

NOTE

The LPCMP's internal DAC output is available as an on-chip internal signal only and is not available for an external chip pin.

ANMUX allows you to select an analog input signal from among eight channel options. One channel option is the DAC output. Other chip resources are connected to the other channels. See the Chip-specific LPCMP information section for more information. ANMUX can operate across the full range of the supply voltage.

44.2.1 Block diagram

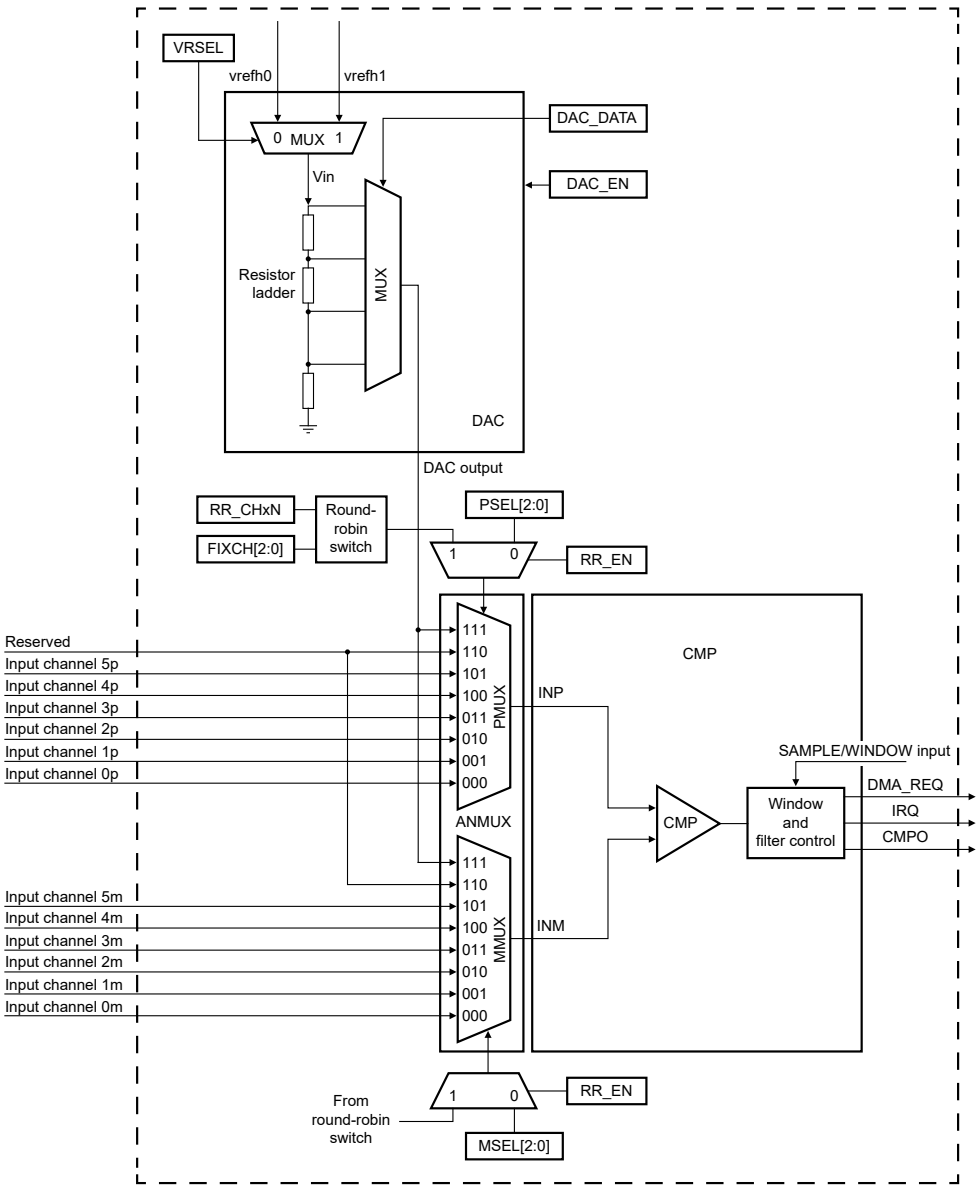


Figure 188. Block diagram

44.2.2 Features

The features of the LPCMP module include：

- Includes two 8-to-1 channel MUXes to select input signal from eight channels
- Supports multiple operation modes to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered
- Provides the following advance features for window and sample:

- Window and sample signals can be inverted.
- COUT (comparator output) rising, falling or both edges closes the window.
- COUT level can be defined when window is closed.

- Provides selectable performance levels:
  - Nano-Power mode
  - Low-Power (speed) mode
  - High-Power (speed) mode
- Supports programmable hysteresis control
- Provides a selectable inversion on comparator output
- Uses an external hysteresis at the same time the output filter is used for internal functions
- Provides interrupt and DMA support
- Supports Round Robin Trigger mode
- Includes an 8-bit resolution DAC
- Provides a selectable supply reference source for DAC
- Provides a configurable Low-Power (speed) mode or High-Power (speed) mode for DAC

44.3 Functional description

44.3.1 Functional block diagram

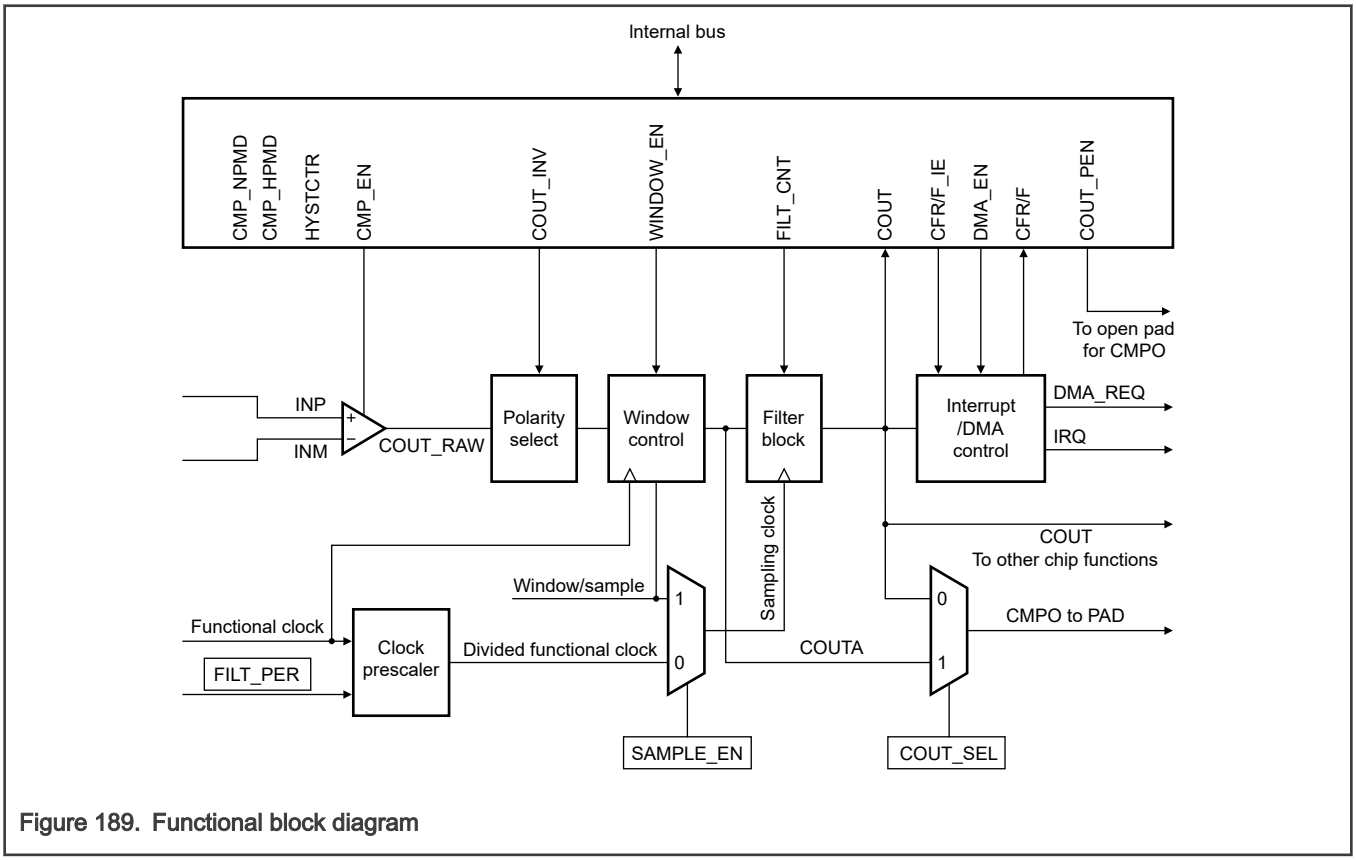


Figure 189. Functional block diagram

As shown in the block diagram, the functions are:

- Compared two analog input voltages applied to INP and INM, COUT_RAW is high when the INP input voltage is greater than the INM input voltage, and COUT_RAW is low when the INP input voltage is less than the INM input voltage.
- The COUT_RAW signal can be inverted by enabling CCR1[COUT_INV].
- The optionally inverted comparator output COUT_RAW is sampled on every functional clock when you enable the CCR1[WINDOW_EN] to generate COUTA. In this case, the comparator output is ignored during time periods when the input voltages are not valid. This is useful when you implement zero-crossing-detection for certain PWM applications.
- The window control block is bypassed when CCR1[WINDOW_EN] is disabled.
- The filter block acts as a simple sampler when CCR1[FILT_CNT] is set to 01h.
- The filter block acts as a filter based on multiple samples when CCR1[FILT_CNT] is set to be greater than 01h.
  - If CCR1[SAMPLE_EN] is set to 1, use the external SAMPLE input as the sampling clock.
  - If CCR1[SAMPLE_EN] is set to 0, use the divided functional clock as the sampling clock.
- Bypasses the filter block when it is not in use.

```
Bypass_Filter_Block = (FILT_CNT == 0x00) |  (~SAMPLE_EN & (FILT_PER == 0x00))
```

- Both COUTA and COUT can be configured as module output CMPO by configuring CCR1[COUT_SEL], and are used for different purposes within the system.
- The optionally filtered COUT can be read directly in CSR[COUT].
- The SAMPLE/WINDOW signal can be inverted by setting CCR1[WINDOW_INV].
- The SAMPLE/WINDOW signal can be closed by COUT's falling edge and/or rising edge by setting CCR1[WINDOW_CLS] in Window mode.
- In Window mode, when window is closed, define the COUTA value as CCR1[COUTA_OW] by setting CCR1[COUTA_OWEN]. If CCR1[COUTA_OWEN] is not set, COUTA holds the last sampled value.

NOTE

See the chip configuration section for the source of SAMPLE/WINDOW input.

44.3.2 Round-robin trigger mode

You can enable Round-Robin Trigger mode by setting RRCR0[RR_EN] to 1. A trigger event initiates a comparison sequence. The next trigger event should not occur before the current sequence completes.

RRCR1[FIXP] and RRCR1[FIXCH] select the reference channel for the plus side mux or the minus side mux. RRCR1[RR_CHnEN] selects active channels.

When a trigger comes, the analog comparator enables. After the comparison sequence completes, the analog comparator disables again. RRCR0[RR_INITMOD] controls the analog stabilization time. Note that RR_INITMOD*round robin clock period must be longer than the initialization delay specified in the Comparator and 8-bit DAC electrical specifications section of LPCMP datasheet.

After the stabilization process completes, the round robin manner comparison sequence begins. Sample the comparison result for the selected active channel after RRCR0[RR_NSAM] defines the configurable number of operation clocks.

There are (RR_SAMPLE_CNT + 1) samples for each channel. With these samples, at least (RR_SAMPLE_THRESHOLD + 1) sampled "1" causes the channel's final sample result to be "1"; otherwise, it is "0". Compare the final result with the pre-programmed value.

After all the active channels are sampled/compared, if the comparison result changes from its pre-programmed state, the corresponding flag in RRSR[RR_CHnF] is set. Write to RRCSR[RR_CHnOUT] to configure the pre-programmed state for each channel. Update RRCSR[RR_CHnOUT] to store the last comparison result for each channel. If any flag in RRSR[RR_CHnF] sets, CSR[RRF] also sets. If IER[RRF_IE] sets, an asynchronous interrupt asserts. Note that these flags do not support generating a DMA transfer event.

The following diagram shows the basic flow of this mode. In the diagram, RRCR1[RR_CH1EN], RRCR1[RR_CH3EN], and RRCR1[RR_CH4EN] are 1, so channels #1, #3, and #4 are selected for round-robin depending on their priority setting. RRCR0[RR_NSAM] sets to 2'b01, so you can sample one clock later the comparison result of the selected channel. After you compare the channel #4, the result is sampled, and round-robin ends. If any of the comparison results from channel #1, #3, or #4 changed from their programmed value (written to RRCSR[RR_CH1OUT], RRCSR[RR_CH3OUT], and RRCSR[RR_CH4OUT]), generates an interrupt . Software can then poll RRSR[RR_CHnF] to see which channel input(s) changed value.

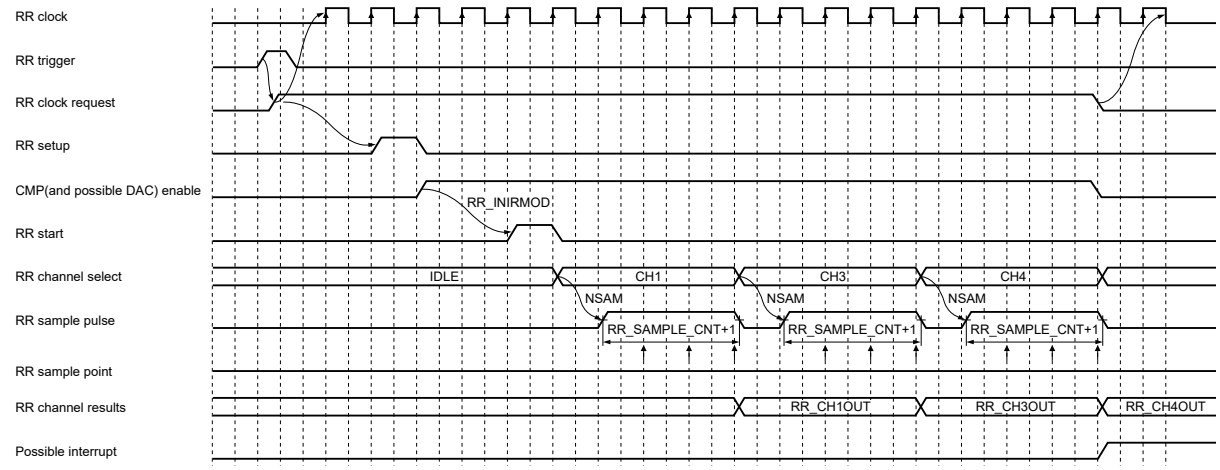


Figure 190. Trigger mode

The table below shows the channel decode in both Functional mode and Trigger mode. Other cases not in the table are illegal.

Table 377. CMP channel decode in functional mode and round-robin trigger mode

Mode	RR_EN	PSEL[2:0]	MSEL[2:0]	FIXP	FIXCH[2:0]	RR_CHxN	INP	INM	CMP Behavior
Functional mode	0	0 to 7	0 to 7	x[1]	x	x	Channel decoded from PSEL[2:0]	Channel decoded from MSEL[2:0]	Compare the channel 0 to 7 with channel 0 to 7[2]
Trigger mode	1	x	x	0	0 to 7	0 to 7	Channel fixed by FIXCH[2:0]	Channel sweep (RR_CHxN)	Sweep the channel 0 to 7 with a fixed channel (0 to 7)
		x	x	1	0 to 7	0 to 7	Channel sweep (RR_CHxN)	Channel fixed by FIXCH[2:0]	Sweep the channel 0 to 7 with a fixed channel (0 to 7)

1. "x" means "don't care"
2. PSEL should not be same as MSEL.

44.3.3 Low-pass filter mode

The low-pass filter mode operates on an unfiltered, optionally inverted comparator output COUTA, and generates the filtered and synchronized output COUT. You can configure both COUTA and COUT as module outputs and use for different purposes within the system.

Synchronization and edge detection determine the bit values of status register. They also apply to COUT for all sampling and windowed modes. You can perform filtering using an internal timebase defined by CCR1[FILT_PER], or use an external sample input to determine sample time.

The need for digital filtering and the amount of filtering depends on your requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, generate a high-frequency oscillations at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

44.3.3.1 Enabling low-pass filter mode

You can enable low-pass filter mode by setting the following:

- CCR1[FILT_CNT] > 01h
- CCR1[FILT_PER] to a nonzero value or writing 1 to CCR1[SAMPLE_EN].

If you use the divided functional clock to drive the low-pass filter, it samples COUTA every CCR1[FILT_PER] functional clock cycle.

If CCR1[SAMPLE_EN] is set to 1, the low-pass filter samples COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CCR1[FILT_CNT] samples agree that the output value has changed.

44.3.3.2 Latency issues

Program the value of CCR1[FILT_PER] or sample period such that the sampling period is longer than the period of the expected noise, ensuring that a given noise spike corrupts only one sample. You must choose the value of CCR1[FILT_CNT] to reduce the probability of noisy samples causing an incorrect transition to recognize. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CCR1[FILT_CNT].

You must trade off the values of CCR1[FILT_PER] or sample period and CCR1[FILT_CNT] against the need for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CCR1[FILT_CNT].

Table 379 summarizes maximum latency values for the various modes of operation in the absence of noise. Filtering latency restarts each time the noise masks an actual output transition.

44.3.4 Low power mode operation

Below table introduces the mode of operation of lower power.

Table 378. Low power mode operation

Mode of operation	Description
Wait	LPCMP can operate normally with window and filter function.
Deep sleep	LPCMP can operate only in Continuous mode or Round-robin trigger mode.

44.3.5 Functional modes

You can combine the comparator window and filter features as shown in the following table.

Table 379. Functional modes

Mode #	CMP_EN	WINDOW_EN	SAMPLE_EN	FILT_CNT	FILT_PER	Operation	Maximum latency[1]
1	0	X	X	X	X	See the Disabled mode (#1).	N/A

Table continues on the next page...

Table 379. Functional modes (continued)

Mode #	CMP_EN	WINDOW_EN	SAMPLE_EN	FILT_CNT	FILT_PER	Operation	Maximum latency[1]
2A	1	0	X	0x00	X	See the Continuous mode (#2A and #2B).	T_{PD}
2B	1	0	0	X	0x00		
3A	1	0	1	0x01	X	See the Sampled, non-filtered mode (#3A and #3B).	$T_{PD} + T_{SAMPLE} + T_{per}$
3B	1	0	0	0x01	> 0x00		$T_{PD} + (FILT_PER * T_{per}) + T_{per}$
4A	1	0	1	> 0x01	X	See the Sampled, filtered mode (#4A and #4B).	$T_{PD} + (FILT_CNT * T_{SAMPLE}) + T_{SAMPLE}$
4B	1	0	0	> 0x01	> 0x00		$T_{PD} + (FILT_CNT * FILT_PER \times T_{per}) + T_{per}$
5A	1	1	0	0x00	X	See the Windowed mode (#5A and #5B).	$T_{PD} + 2T_{per}$
5B	1	1	0	X	0x00		
6	1	1	0	0x01	> 0x00	See the Windowed/ Resampled mode (#6).	$T_{PD} + (FILT_PER * T_{per}) + 3T_{per}$
7	1	1	0	> 0x01	> 0x00	See the Windowed/ Filtered mode (#7).	$T_{PD} + (FILT_CNT * FILT_PER \times T_{per}) + 3T_{per}$
All other combinations of CMP_EN, WINDOW_EN, SAMPLE_EN, FILT_CNT, and FILT_PER are illegal.							

1. T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{SAMPLE} is the clock period of the external sample clock. T_{per} is the period of the functional clock.

44.3.5.1 Disabled mode (#1)

In this mode:

- The analog comparator is non-functional and consumes no power.
- CSR[COUT] and CMPO are the same as CCR1[COUT_INV].

44.3.5.2 Continuous mode (#2A and #2B)

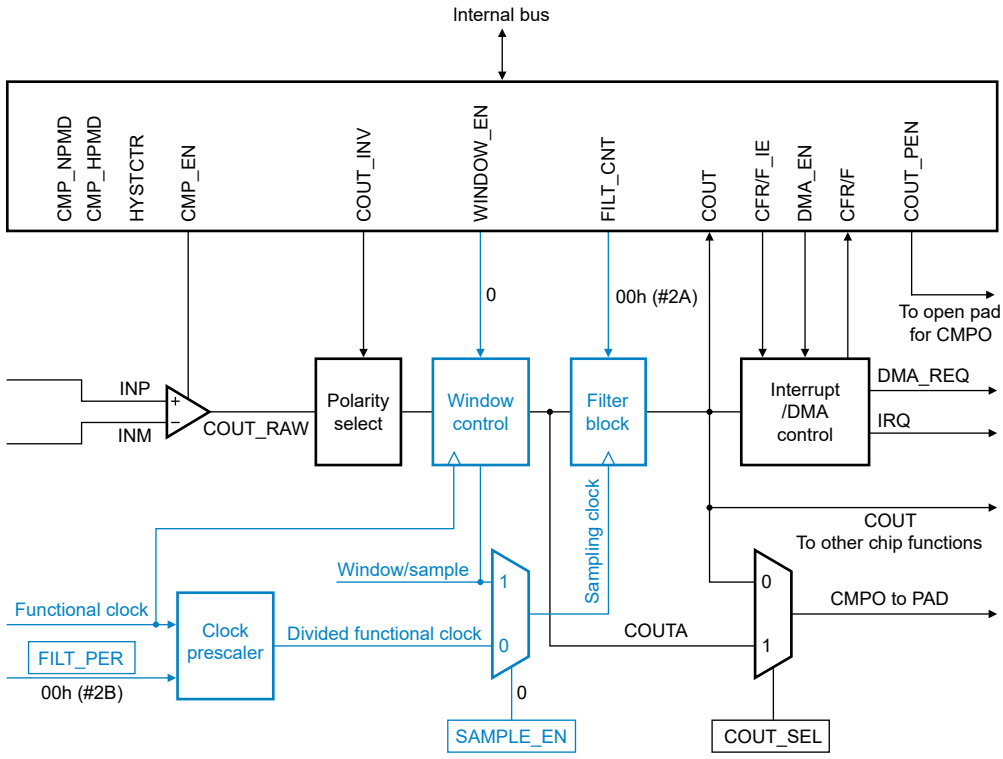


Figure 191. Comparator operation in continuous mode

COUT_RAW is optionally inverted in this mode but is not subject to external sampling or filtering. Both window control and filter blocks bypass completely, and CSR[COUT] updates continuously. The path from comparator input pins to output pins operates in a combinational (unclocked) mode. COUT and COUTA are identical in this mode.

For cases where a comparator drives a fault input, you must configure it to operate in Continuous mode so that an external fault can immediately pass to the target fault circuitry through the comparator.

44.3.5.3 Sampled, non-filtered mode (#3A and #3B)

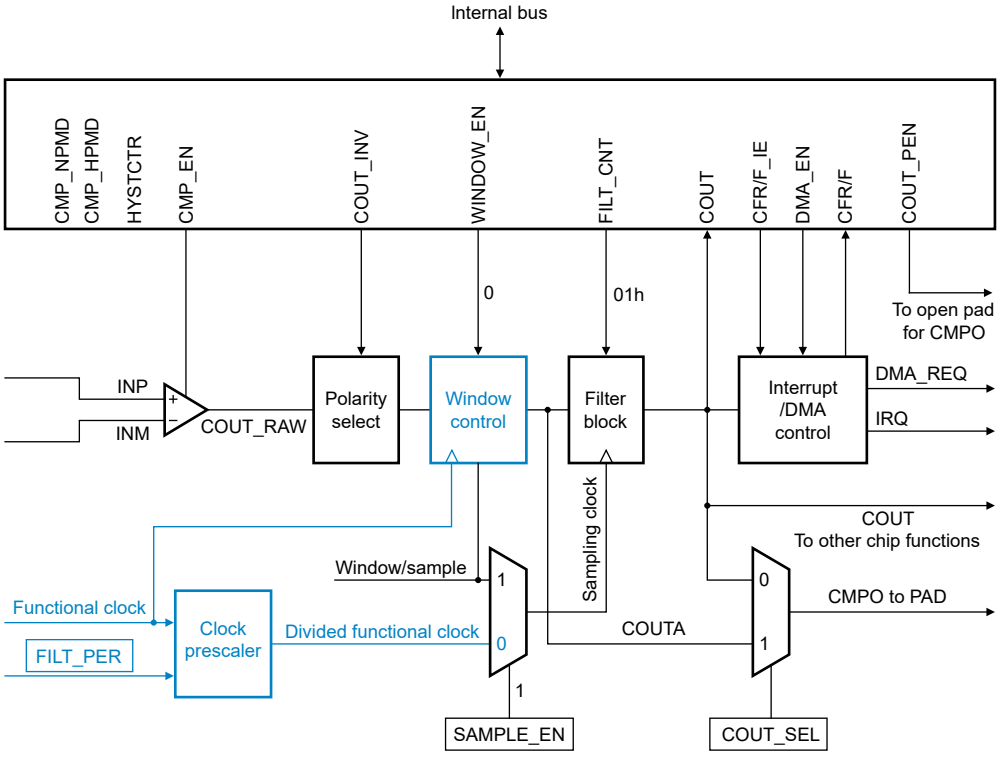


Figure 192. Sampled, non-filtered (#3A): sampling point externally driven

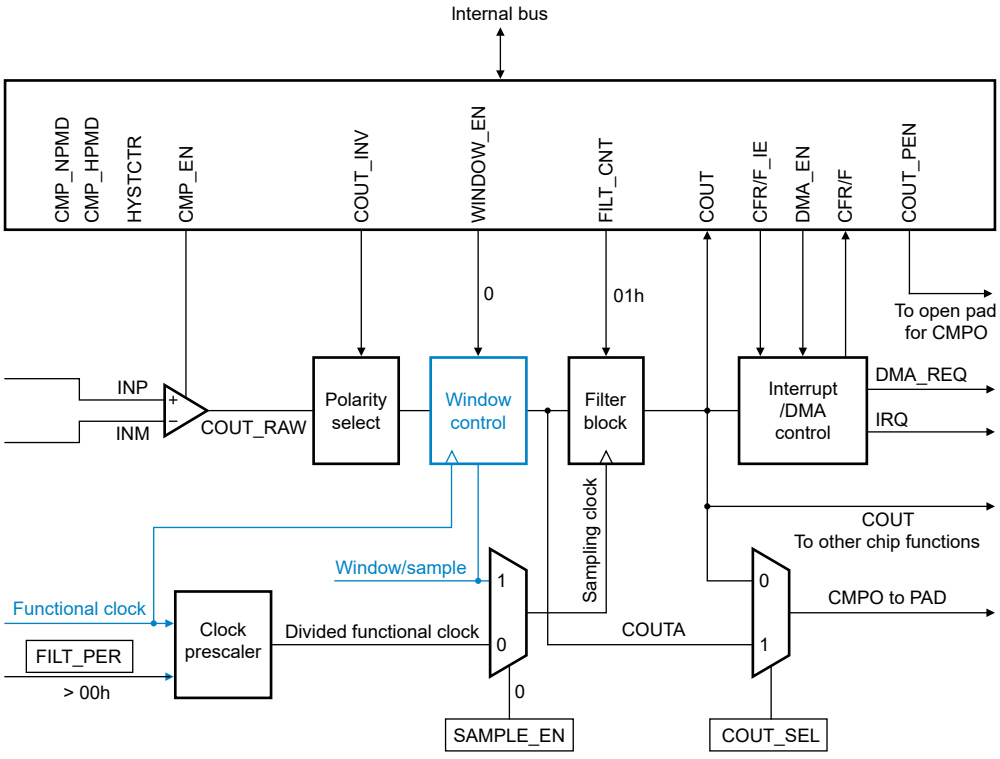


Figure 193. Sampled, Non-Filtered (#3B): sampling interval internally derived

In this mode, the path from analog inputs to COUTA is combinational (unclocked). Windowing control bypasses completely. You can sample COUTA whenever you detect a rising edge on the sampling clock.

The difference in two operation modes (#3A and #3B) of sampled, non-filtered mode is that how you drive the clock to the filter block. In #3A, the clock to filter block drives externally, and in #3B, the clock to filter block drives internally.

The filter block has no other function than sample or hold of the comparator output in this mode.

The following figure shows the comparator operation in this mode, assuming that the polarity select sets to a non-inverting state.

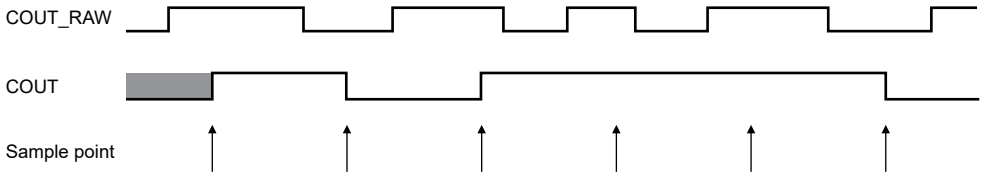


Figure 194. Sampled, Non-Filtered mode timing diagram

44.3.5.4 Sampled, filtered mode (#4A and #4B)

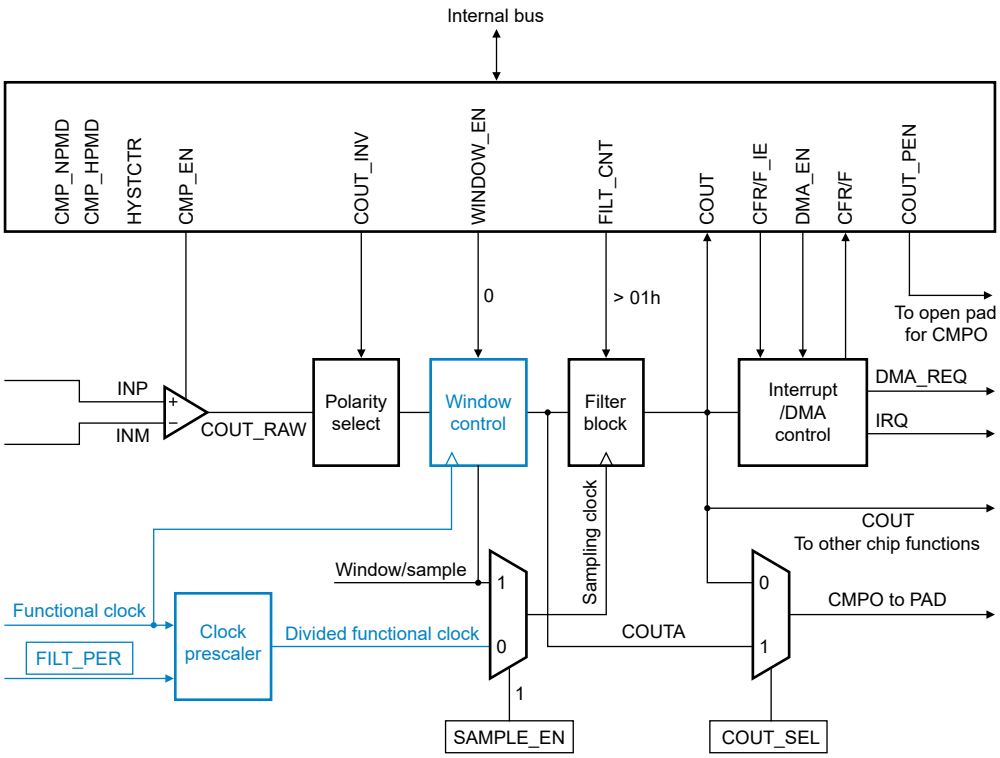


Figure 195. Sampled, filtered (#4A): sampling point externally driven

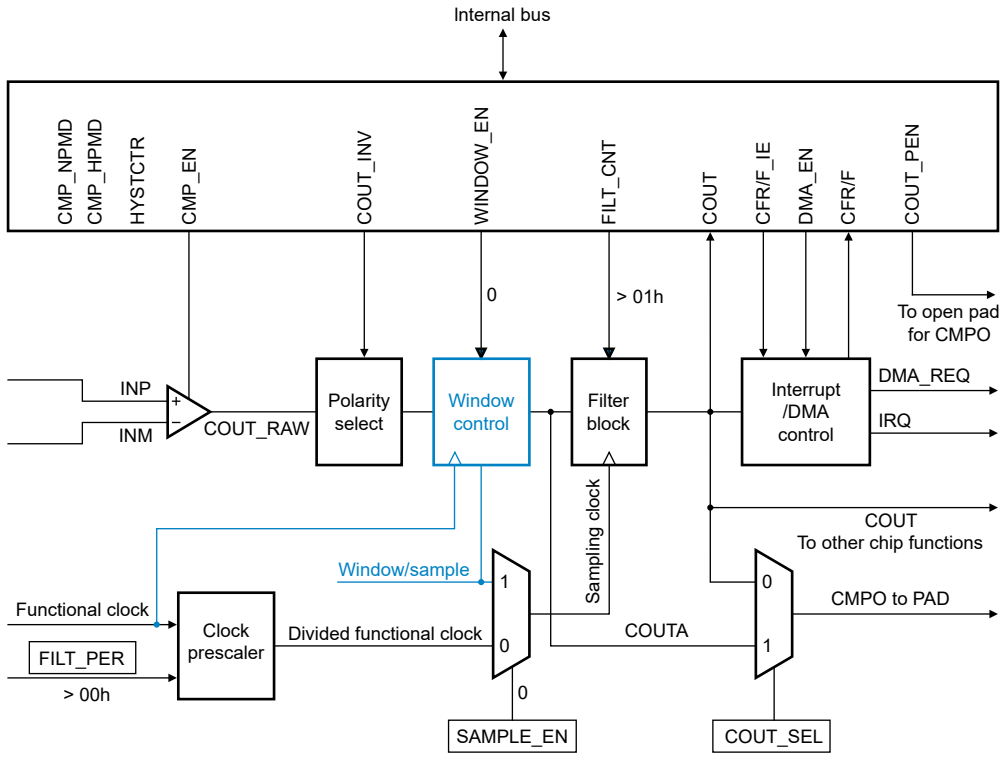


Figure 196. Sampled, filtered (#4B): sampling point internally derived

In this mode, the path from the analog inputs to COUTA is combinational(unclocked). Windowing control bypasses completely. You can sample COUTA whenever you detect a rising edge on the sampling clock.

The only difference in operation between sampled, non-filtered (#3A) mode and sampled, filtered (#4A) mode is that CCR1[FILT_CNT] is larger than 1, which activates filter operation.

The only difference in operation between sampled, non-filtered (#3B) mode and sampled, filtered (#4B) mode is that CCR1[FILT_CNT] is larger than 1, which activates filter operation.

44.3.5.5 Windowed mode (#5A and #5B)

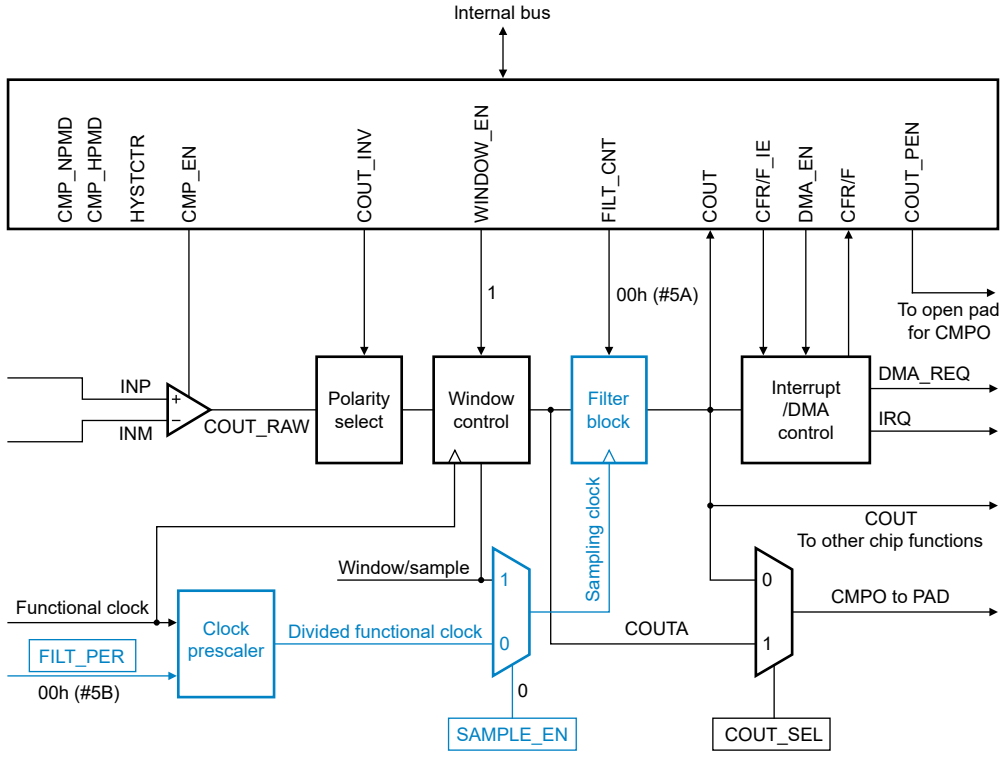


Figure 197. Windowed mode

The functional clock clocks COUTA whenever you enable the window in this mode. The last latched value holds after you disable the window and the filter block is bypassed.

The following figure shows the comparator operation in this mode, ignoring the latency of the analog comparator, polarity select, and window control block. The polarity select sets to a non-inverting state.

COUTA may lag the analog inputs by up to two functional clock cycles plus the combinational path delay through the comparator and polarity select logic in the actual operation.

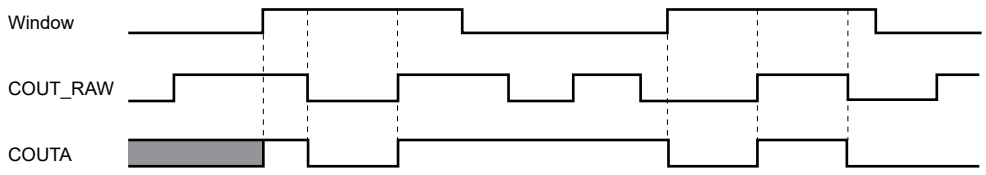


Figure 198. Windowed mode timing diagram

The following figure shows that if CCR1[COUTA_OWEN] becomes 1, you can define COUTA level as CCR1[COUTA_OW], after you closes the window.

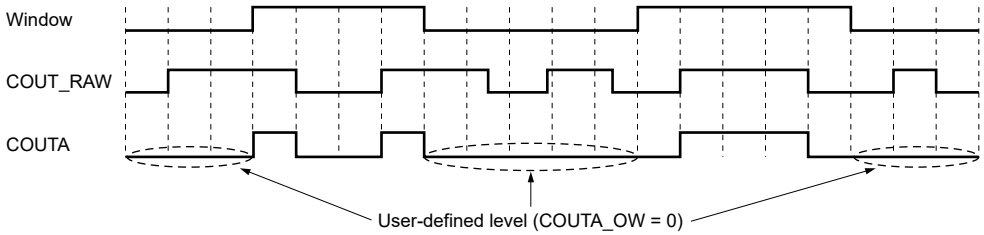


Figure 199. Windowed mode timing diagram with user defined value 0 outside window

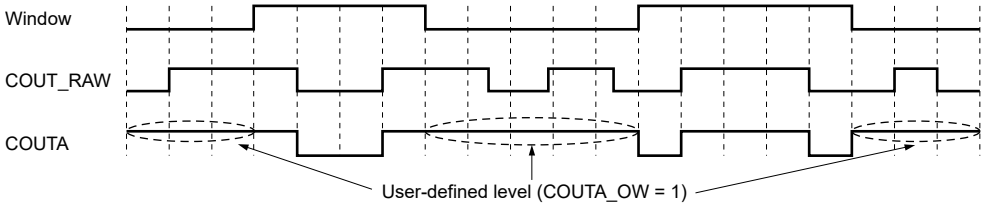


Figure 200. Windowed mode timing diagram with user defined value 1 outside window

If CCR1[WINDOW_CLS] becomes 1, you can define the COUT event (rising edge, falling edge or both edges that CCR1[EVT_SEL] selects) to close the window. The external window signal has to go to zero and back to one to enable the internal window again. The following figure shows an example that COUT rising edge closes the internal window.

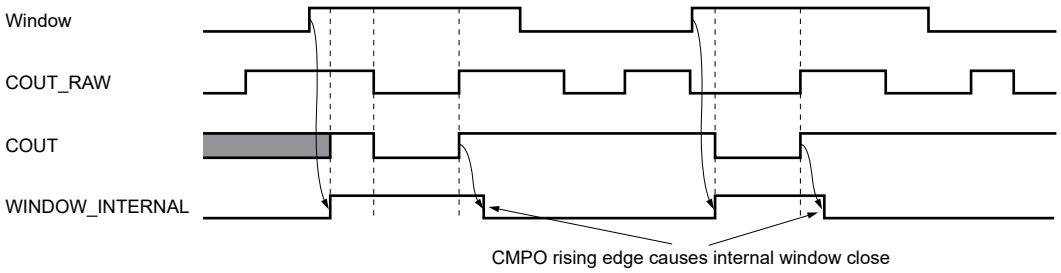


Figure 201. Windowed mode timing diagram with COUT rising edge close window

The following figure shows that if CCR1[WINDOW_INV] becomes 1, you can invert the window signal before you use it.

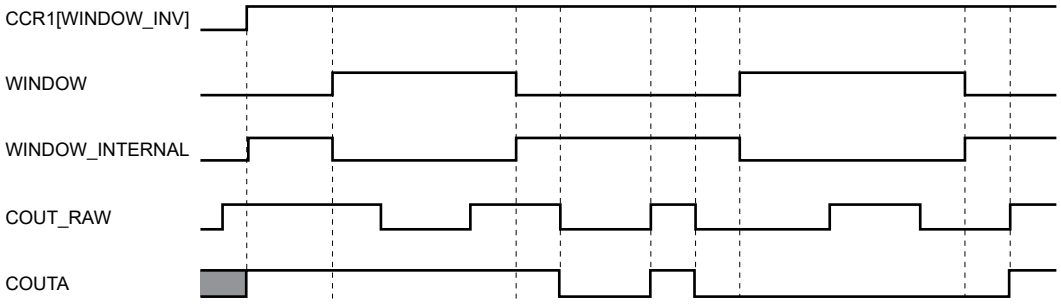


Figure 202. Windowed mode timing diagram with window signal inverted

44.3.5.6 Windowed/Resampled mode (#6)

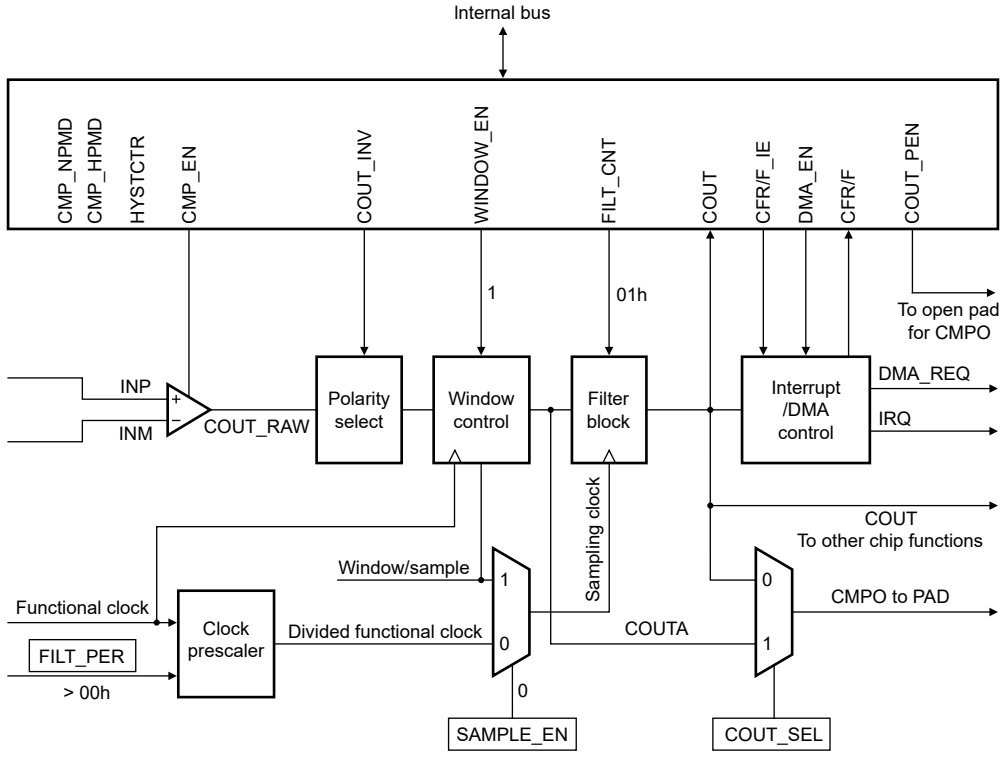


Figure 203. Windowed/Resampled mode

This mode of operation results in an unfiltered string of comparator samples where CCR1[FILT_PER] and the functional clock rate determines the interval between the samples. The following section shows that the configuration for this mode is virtually identical to that for the Windowed/Filtered mode. The only difference is that the value of CCR1[FILT_CNT] must be 1 in this mode.

The following figure uses the same input stimulus shown in Figure 198, and adds resampling of COUTA to generate COUT. The arrows in the figure indicate the time points at which the samples are taken. You can ignore prop delays and latency for clarity.

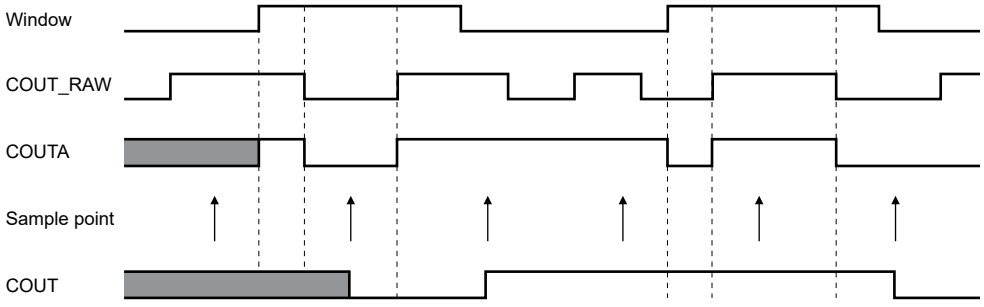


Figure 204. Windowed/Resampled mode operation

This example demonstrates the operation of the comparator in Windowed/Resampled mode, and does not reflect any specific application. Based on the sampling rate and window placement, COUT may not see zero-crossing events that the analog comparator detects. You must carefully consider the sampling period and/or window placement for a given application.

44.3.5.7 Windowed/Filtered mode (#7)

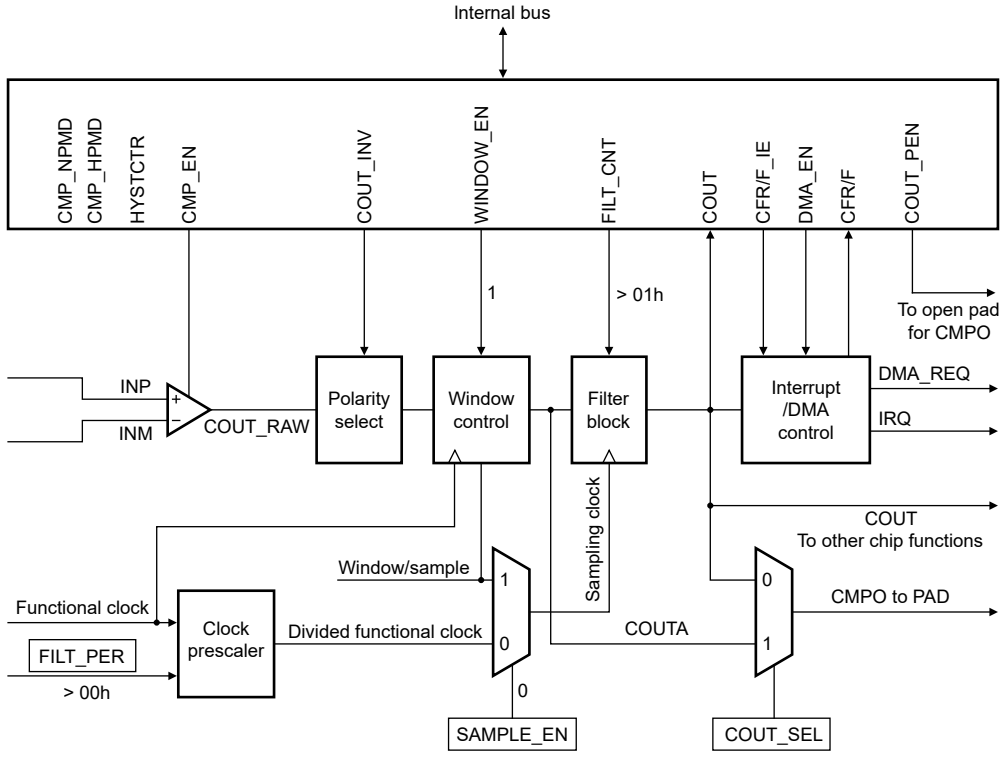


Figure 205. Windowed/Filtered mode

The only difference in operation between Windowed/Resampled mode (#6) and Windowed/Filtered mode (#7) is that CCR1[FILT_CNT] is >1, which activates filter operation.

This mode is the most complex mode of operation for the comparator block, as it utilizes both windowing and filtering features. It also has the highest latency of any of the modes. This is approximately: up to 2 peripheral clock synchronization in the window function + ((CCR1[FILT_CNT] x CCR1[FILT_PER]) + 1) x peripheral clock for the filter function.

44.3.6 DMA

After DMA is enabled by writing 1 to CCR1[DMA_EN] and interrupt is enabled by writing 1 to IER[CFR_IE], IER[CFF_IE], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt. After the DMA completes the transfer, it sends a transfer completing indicator signal that deasserts the DMA transfer request and clears the flags (both CSR[CFR] and CSR[CFF]) to allow a subsequent change on comparator output to occur and forces another DMA request.

A DMA transfer request wakes up the system from Deep sleep mode. After completing the data transfer, the system go back again to Deep sleep mode. See the DMA chapters in this document for more information on the asynchronous DMA function.

44.3.7 Clocking

LPCMP requires the following clocks to operate:

Table 380. LPCMP clocks

Type of clock	Description
Bus	Controls the access to LPCMP registers.

Table continues on the next page...

Table 380. LPCMP clocks (continued)

Type of clock	Description
Functional	Controls window/filter function. LPCMP configures CCR1[FUNC_CLK_SEL] to select one from the four functional clock sources. See the Chip-specific LPCMP information for more on the functional clock source information.
Round-robin clock (RCLK)	Controls Round-robin trigger mode. LPCMP configures CCR1[RR_CLK_SEL] to select one from the four round-robin clock sources. See the Chip-specific LPCMP information for more on the round robin clock source information.

44.3.8 Resets

The global chip reset signal resets LPCMP.

44.3.9 Interrupts

After the corresponding IER becomes 1, CSR[CFR], CSR[CFF], and CSR[RRF] can generate an interrupt, assuming that CCR1[DMA_EN] is not 1. You can clear either the flag or IER to deassert the interrupt.

44.4 External signal descriptions

Below table introduces external signals.

Table 381. External signal descriptions

Signal	Description	I/O
CMPO	Filtered or unfiltered comparator output	O
Input_Analog_Channels	Analog input channels (see the chip-specific information for more on the connections).	I
VREFH_EXT	External reference voltage for the CMP-DAC (see the chip-specific information for more on the connections).	I
RR_ACTIVE	Round-robin trigger mode enabled.	O

44.5 Initialization

You can enable LPCMP by writing 1 to CCR0[CMP_EN], and then configuring the control registers (CCR1, CCR2, DCR, and so on).

To disable LPCMP, write 0 to CCR0[CMP_EN]. Switching operation modes or changing control register fields on-the-fly (when CCR0[CMP_EN] is set to 1) may cause noise on the COUT or COUTA signals. To avoid unwanted signal noise, you must ensure to disable the module before switching modes or changing control fields.

The time required to stabilize COUT is the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function, and filter (see the Comparator and 8-bit DAC electrical specifications section of LPCMP datasheet for more information on propagation delay and power-up delay). Table 379 specifies the delay that the windowing and filter function causes.

During operation, you must always consider the propagation delay of the selected data paths. It can take many functional clock cycles for COUT and CSR[CFR]/CSR[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.

44.6 Application information

44.6.1 Round-robin trigger mode programing recommendation

Configure the Round-robin trigger mode as follows:

1. Configure RRCR0[RR_CLK_SEL] to select the RR clock source.
2. Configure the comparison cycles by RRCR0[RR_NSAM]. Note: It is a mandatory request that the round robin cycling period must set longer than the time that all the active channels complete the specified comparison cycles set by RRCR0[RR_NSAM].
3. Configure CMP initialization delay by RRCR0[RR_INITMOD]. Note: In programming RRCR0[RR_INITMOD], the RR_INITMOD x round robin clock period must be longer than the initialization delay, see the LPCMP datasheet for more information.
4. Configure RRCR0[RR_SAMPLE_CNT] and RRCR0[RR_SAMPLE_THRESHOLD].
5. Configure RRCR0[RR_TRG_SEL] to select the trigger from internal or external.
6. Enable RRCR2[RR_TIMER_EN] and configure RRCR2[RR_TIMER_RELOAD] according to the round robin clock frequency, if using an internal trigger.
7. Configure RRCR1[FIXP] to select the fixed port of CMP and RRCR1[FIXCH] to select the fixed channel.
8. Configure channels for comparison by RRCR1[RR_CHnEN].
9. Write RRCSR[RR_CHnOUT] to define the pre-set state of channel n.
10. Clear channel flags RRSR[RR_CHnF].
11. Enable round robin interrupt by IER[RRF_IE] (disable IER[CFR_IE] and IER[CFF_IE]).
12. Enable round-robin trigger mode by setting RRCR0[RR_EN] to 1.

44.6.2 Round-robin clock (RCLK) frequency requirement

(1) RCLK high frequency limit

RCLK high frequency limit depends on two facts:

1. The analog CMP and DAC initialization time (see the chip data sheet for more information on the initialization time.)
   - RRCR0[RR_INITMOD] provides a maximum 63 RCLK cycles for the analog CMP and DAC initialization.
   - RCLK must be slow to assure: $63 * (1/f_{RCLK}) > T_{initialization}$, where f_{RCLK} is in MHz, and $T_{initialization}$ is in microsecond.
   - so $f_{RCLK} < 63 / T_{initialization}$
   - Example: $T_{initialization}$ = 40 µs, then f_{RCLK} should be smaller than 1.575 MHz.
2. The analog CMP propagation delay (see the Comparator and 8-bit DAC electrical specifications section of LPCMP datasheet for more information on the CMP propagation delay.)
   - RRCR0[RR_NSAM] provides a maximum 4 RCLK cycles for the analog CMP propagation delay.
   - RCLK must be slow to assure: $4 * (1/f_{RCLK}) > T_{propagation}$, where f_{RCLK} is in MHz, and $T_{propagation}$ is in microsecond.
   - $f_{RCLK} < 4 / T_{propagation}$
   - Example: $T_{propagation}$ = 0.1 µs, then f_{RCLK} must be smaller than 40 MHz.

(2) RCLK low frequency limit

In theory, RCLK frequency has no low limit. But the lower the RCLK frequency, the longer the scan time. Therefore, the lower limit of the RCLK frequency depends on the system application.

44.7 LPCMP register descriptions

The memory map comprises of 32-bit aligned registers, which you can access via 8-, 16- or 32-bit reads and 32-bit write. Attempted accesses using unsupported write data sizes, writes to read-only resources, or to reserved spaces terminate with an error. Read access to reserved address generates a transfer error and the read data bus shows all 0s.

44.7.1 LPCMP memory map

CMP0 base address: 4005_1000h

CMP1 base address: 4005_2000h

CMP2 base address: 4005_3000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0100_0001h
4h	Parameter (PARAM)	32	R	0000_0002h
8h	Comparator Control Register 0 (CCR0)	32	RW	0000_0002h
Ch	Comparator Control Register 1 (CCR1)	32	RW	0000_0000h
10h	Comparator Control Register 2 (CCR2)	32	RW	0000_0000h
18h	DAC Control (DCR)	32	RW	0000_0000h
1Ch	Interrupt Enable (IER)	32	RW	0000_0000h
20h	Comparator Status (CSR)	32	RW	0000_0000h
24h	Round Robin Control Register 0 (RRCR0)	32	RW	0000_0000h
28h	Round Robin Control Register 1 (RRCR1)	32	RW	0000_0000h
2Ch	Round Robin Control and Status (RRCSR)	32	RW	0000_0000h
30h	Round Robin Status (RRSR)	32	RW	0000_0000h
38h	Round Robin Control Register 2 (RRCR2)	32	RW	0000_0000h

44.7.2 Version ID (VERID)

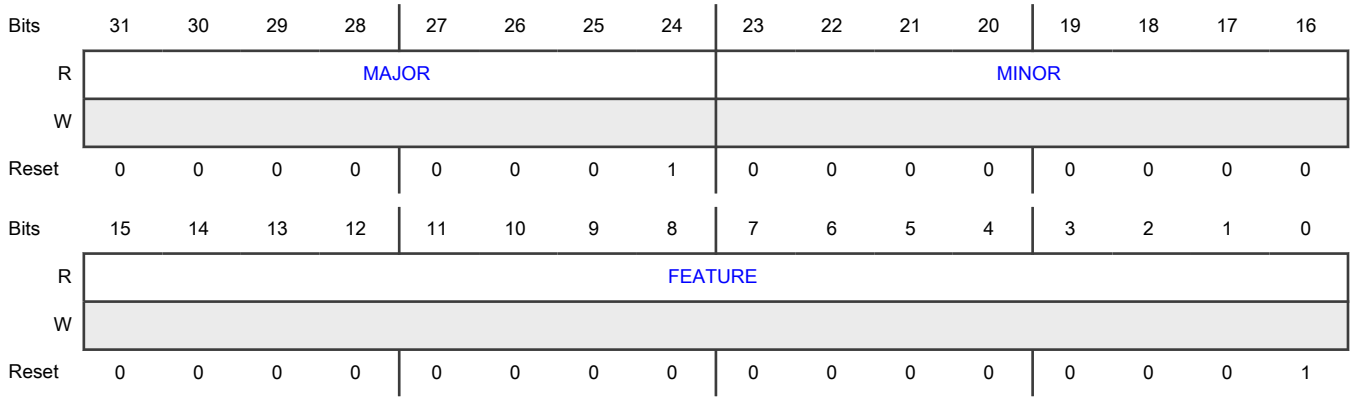
Offset

Register	Offset
VERID	0h

Function

Contains version numbers for the module design and feature set.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module design.
23-16 MINOR	Minor Version Number Returns the minor version number for the module design.
15-0 FEATURE	Feature Specification Number Returns the feature set number. 0000_0000_0000_0001b - Round robin feature

44.7.3 Parameter (PARAM)

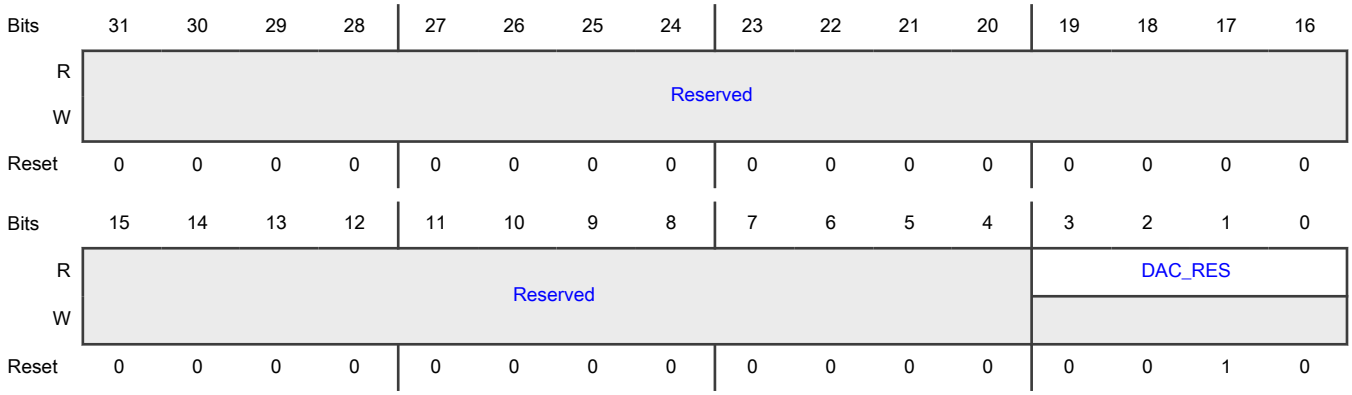
Offset

Register	Offset
PARAM	4h

Function

Contains parameter values that are implemented in the module.

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 DAC_RES	DAC Resolution Indicates supported DAC resolutions. **NOTE** All other bit field values are reserved. 0000b - 4-bit DAC 0001b - 6-bit DAC 0010b - 8-bit DAC 0011b - 10-bit DAC 0100b - 12-bit DAC 0101b - 14-bit DAC 0110b - 16-bit DAC

44.7.4 Comparator Control Register 0 (CCR0)

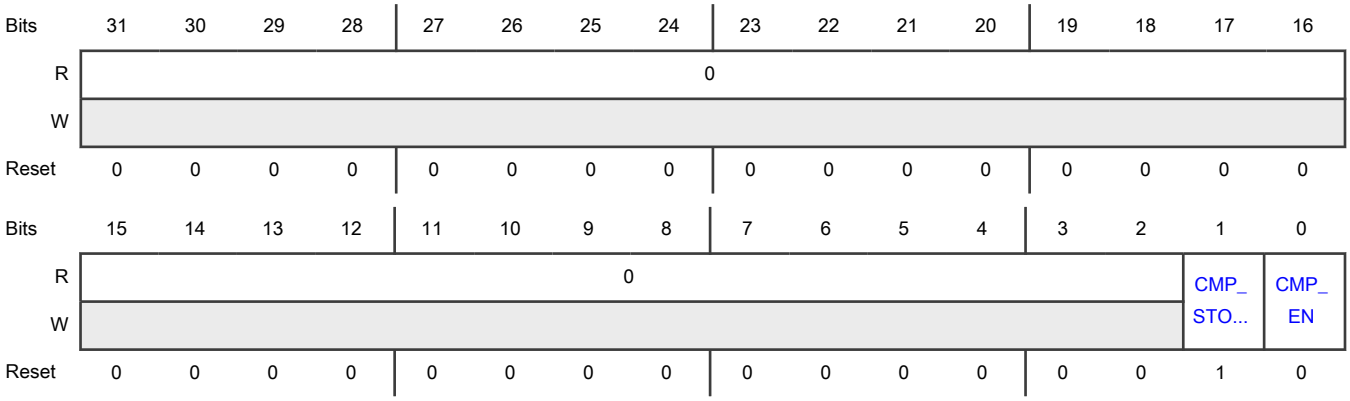
Offset

Register	Offset
CCR0	8h

Function

Contains configuration options for enabling the analog comparator.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 CMP_STOP_EN	Comparator Deep sleep Mode Enable Enables the analog comparator or the DAC when the module is in Deep sleep mode. **NOTE** This field has no effect in Round-robin Trigger mode. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
CMP0	—	CCR0
CMP1	—	CCR0
CMP2	CCR0	—

0b - Disables the analog comparator regardless of CMP_EN.

1b - Allows CMP_EN to enable the analog comparator.

0 CMP_EN	Comparator Enable Enables the analog comparator. 0b - Disable (The analog logic remains off and consumes no power.) 1b - Enable

44.7.5 Comparator Control Register 1 (CCR1)

Offset

Register	Offset
CCR1	Ch

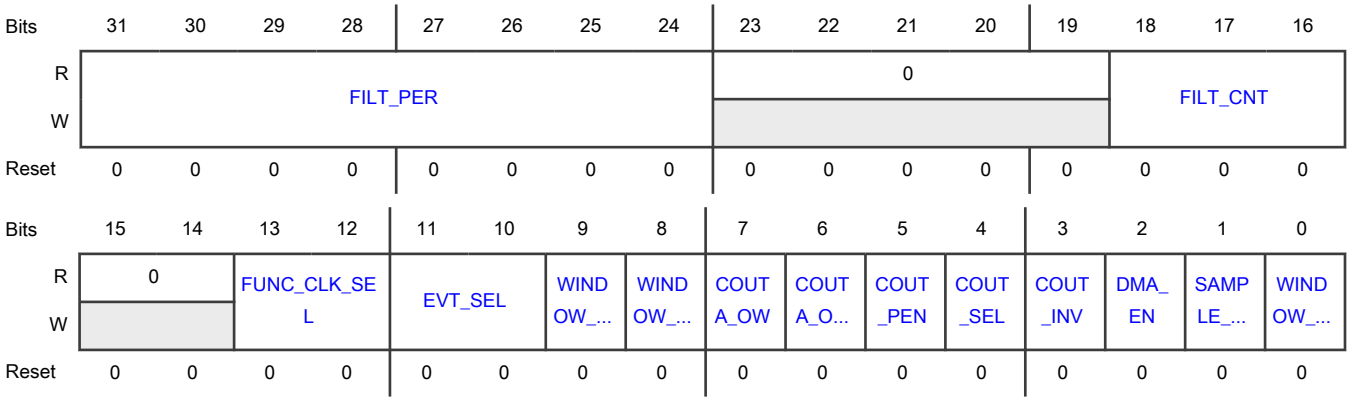
Function

Contains configuration options for the comparator operation, such as enabling Sampling or Windowing mode.

NOTE

You cannot enable Sampling and Windowing modes both at the same time. Sampling mode takes precedence over Windowing mode. If you write 1 to both SAMPLE_EN and WINDOW_EN, only SAMPLE_EN becomes 1.

Diagram



Fields

Field	Function
31-24 FILT_PER	Filter Sample Period Specifies the sampling period (in functional clock cycles) of the comparator output filter. Programming this field to 00h bypasses the filter. See Functional description for more information on filter programming and latency. **NOTE** FILT_PER has no effect in Sampling mode (CCR1[SAMPLE_EN] = 1).
23-19 —	Reserved
18-16 FILT_CNT	Filter Sample Count Specifies the number of consecutive samples that must agree before the comparator output filter accepts the sample as a new valid output state. See Functional description for more information on filter programming and latency.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000b - Filter is bypassed: COUT = COUTA 001b - 1 consecutive sample (Comparator output is simply sampled.) 010b - 2 consecutive samples 011b - 3 consecutive samples 100b - 4 consecutive samples 101b - 5 consecutive samples 110b - 6 consecutive samples 111b - 7 consecutive samples
15-14 —	Reserved
13-12 FUNC_CLK_SEL	Functional Clock Source Select Selects which clock source is used for the functional clock. See the Chip-specific LPCMP information for more on the functional clock source information. 00b - Select functional clock source 0 01b - Select functional clock source 1 10b - Select functional clock source 2 11b - Select functional clock source 3
11-10 EVT_SEL	COUT Event Select Selects which COUT signal edge (rising, falling, or both) defines a COUT event. **NOTE** Valid only in Windowing mode. 00b - Rising edge 01b - Falling edge 1xb - Both edges
9 WINDOW_CLS	COUT Event Window Close Enables a COUT event (defined as a COUT rising edge, falling edge, or both) to close an active window. See EVT_SEL to configure the COUT event. **NOTE** The WINDOW signal has to go to zero and back to one again to re-activate the window. Valid only in Windowing mode. 0b - COUT event cannot close the window 1b - COUT event can close the window
8	WINDOW/SAMPLE Signal Invert

Table continues on the next page...

Table continued from the previous page...

Field	Function
WINDOW_INV	Inverts the window/sample signal. 0b - Do not invert 1b - Invert
7 COUTA_OW	COUTA Output Level for Closed Window Defines the COUTA signal value when the window is closed. **NOTE** Valid only in Windowing mode and when COUTA_OWEN=1. 0b - COUTA is 0 1b - COUTA is 1
6 COUTA_OWEN	COUTA_OW Enable Enables the COUTA signal value to be defined by COUTA_OW when the window is closed. **NOTE** Valid only in Windowing mode. 0b - COUTA holds the last sampled value. 1b - Enables the COUTA signal value to be defined by COUTA_OW.
5 COUT_PEN	Comparator Output Pin Enable Enables the comparator output to become an available signal option for a selected package pin. 0b - Not available 1b - Available
4 COUT_SEL	Comparator Output Select Selects which comparator output option, COUT or COUTA, to use for CMPO. 0b - Use COUT (filtered) 1b - Use COUTA (unfiltered)
3 COUT_INV	Comparator Invert Selects the polarity of the analog comparator function, affecting the value driven to the COUT output (on both the chip pin and as CSR[COUT]) when CCR0[CMP_EN] is 0. **NOTE** COUT_INV has no effect in Trigger mode. 0b - Do not invert 1b - Invert
2	DMA Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DMA_EN	Enables DMA transfers triggered from the LPCMP module. After this field and the corresponding interrupt enable field becomes 1, a DMA request is asserted when CFR or CFF becomes 1. 0b - Disable 1b - Enable
1 SAMPLE_EN	Sampling Enable Enables Sampling mode. 0b - Disable 1b - Enable
0 WINDOW_EN	Windowing Enable Enables Windowing mode. **NOTE** Valid only when SAMPLE_EN = 0. 0b - Disable 1b - Enable

44.7.6 Comparator Control Register 2 (CCR2)

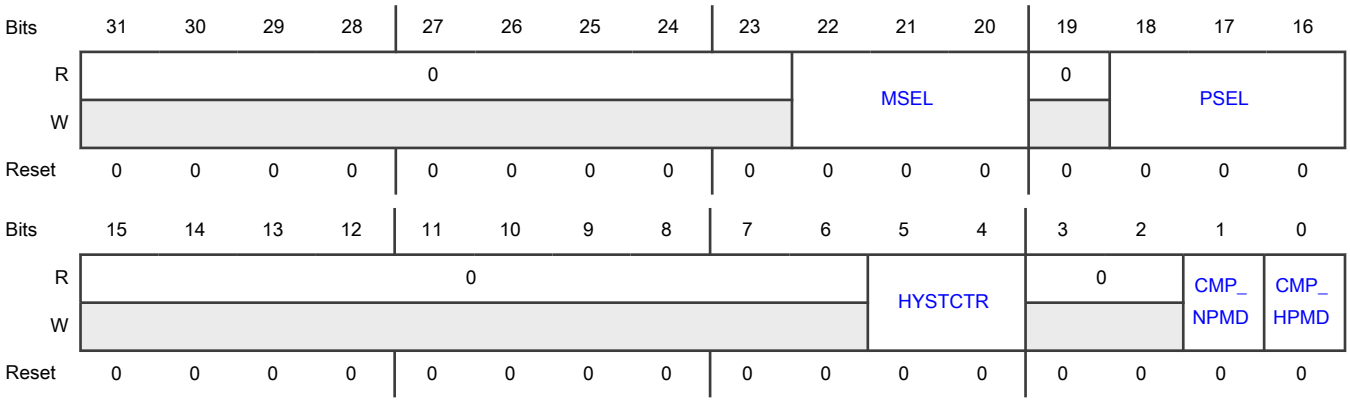
Offset

Register	Offset
CCR2	10h

Function

Contains the configuration options for the comparator operation, such as selecting the plus and minus comparator inputs and the hysteresis levels.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-20 MSEL	Minus Input MUX Select Selects the input used for the negative mux. See the chip-specific LPCMP information for more on connections. **NOTE** MSEL has no effect in Trigger mode. 000b - Input 0m 001b - Input 1m 010b - Input 2m 011b - Input 3m 100b - Input 4m 101b - Input 5m 110b - Reserved 111b - Internal DAC output
19 —	Reserved
18-16 PSEL	Plus Input MUX Select Selects the input used for the positive mux. See the chip-specific LPCMP information for more on connections. **NOTE** PSEL has no effect in Trigger mode. 000b - Input 0p 001b - Input 1p 010b - Input 2p 011b - Input 3p 100b - Input 4p 101b - Input 5p 110b - Reserved 111b - Internal DAC output
15-6 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-4 HYSTCTR	Comparator Hysteresis Control Selects the level of internally generated hysteresis for the comparator output. **NOTE** This applies to the comparator hard block. 00b - Level 0: Analog comparator hysteresis 0 mV. 01b - Level 1: Analog comparator hysteresis 10 mV. 10b - Level 2: Analog comparator hysteresis 20 mV. 11b - Level 3: Analog comparator hysteresis 30 mV.
3-2 —	Reserved
1 CMP_NPMD	CMP Nano Power Mode Select Enables Nano Power mode for the comparator. 0b - Disables CMP Nano power mode. CCR2[CMP_HPMD] determines the mode for the comparator. 1b - Enables CMP Nano power mode.
0 CMP_HPMD	CMP High Power Mode Select Selects Low or High Power(Speed) mode for the comparator. **NOTE** Valid only when not in Nano Power mode (CMP_NPMD = 0). 0b - Low power (speed) comparison mode 1b - High power (speed) comparison mode

44.7.7 DAC Control (DCR)

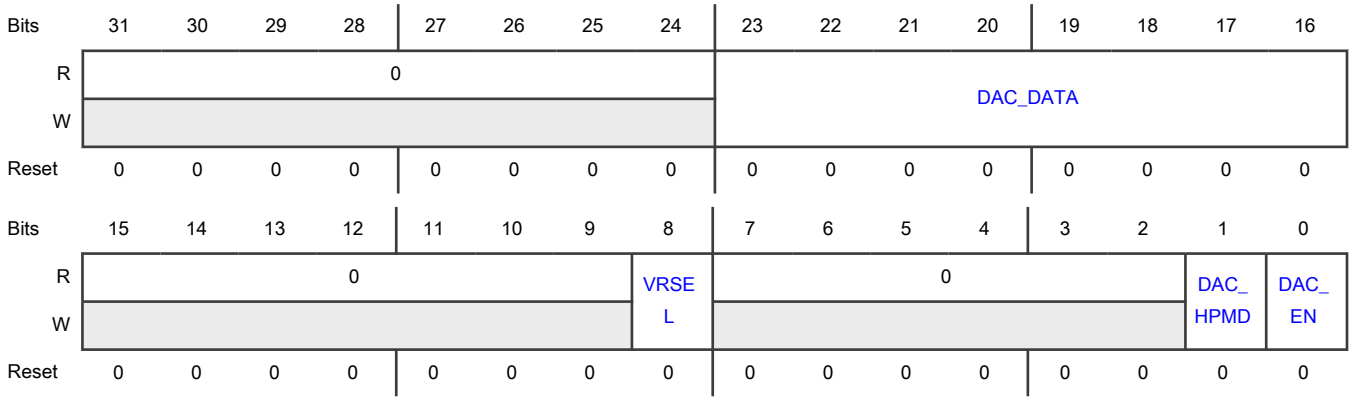
Offset

Register	Offset
DCR	18h

Function

Contains the configuration options to enable the DAC.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 DAC_DATA	DAC Output Voltage Select Selects the DAC output (DACO) voltage from one of 256 distinct levels by configuring the value of DAC_DATA. The DACO ranges from Vin/256 to Vin. **NOTE** DACO = (Vin/256) * (DAC_DATA + 1)
15-9 —	Reserved
8 VRSEL	DAC Reference High Voltage Source Select Selects the high voltage reference source for the Vin supply of the DAC's resistor ladder network. See the chip-specific LPCMP information for the source of vrefh0 and vrefh1. 0b - VREFH0 1b - VREFH1
7-2 —	Reserved
1 DAC_HPMD	DAC High Power Mode Enables the DAC high power mode. 0b - Disable 1b - Enable
0 DAC_EN	DAC Enable Enables the DAC. When disabled, power-down the DAC to conserve power.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable

44.7.8 Interrupt Enable (IER)

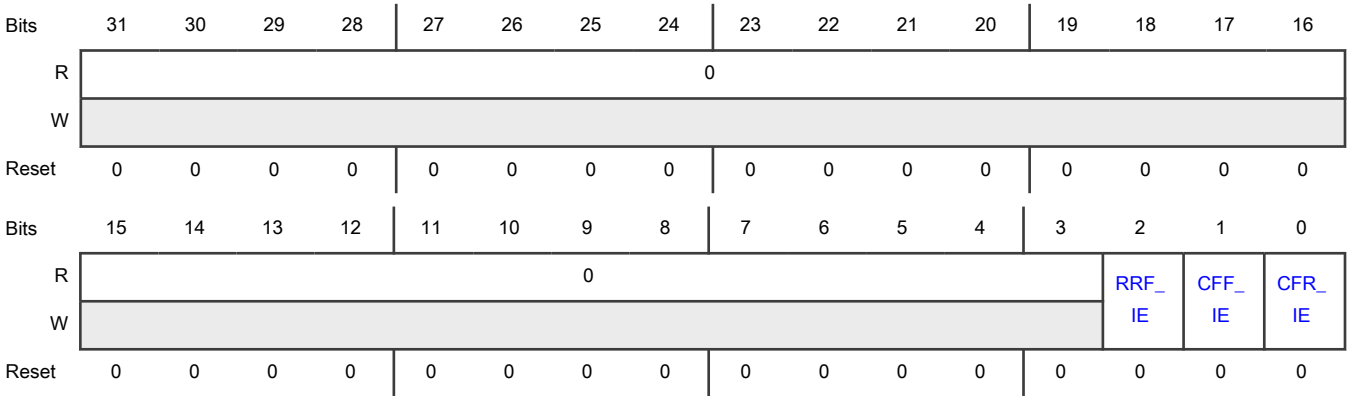
Offset

Register	Offset
IER	1Ch

Function

Provides enable fields for the comparator and round-robin flag interrupts.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 RRF_IE	Round-Robin Flag Interrupt Enable Enables or disables the round-robin flag interrupt. 0b - Disables the round-robin flag interrupt. 1b - Enables the round-robin flag interrupt when the comparison result changes for a given channel.
1	Comparator Flag Falling Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
CFF_IE	Enables or disables the comparator flag falling interrupt. 0b - Disables the comparator flag falling interrupt. 1b - Enables the comparator flag falling interrupt when CFF is set.
0 CFR_IE	Comparator Flag Rising Interrupt Enable Enables or disables the comparator flag rising interrupt. 0b - Disables the comparator flag rising interrupt. 1b - Enables the comparator flag rising interrupt when CFR is set.

44.7.9 Comparator Status (CSR)

Offset

Register	Offset
CSR	20h

Function

Indicates comparator status, including COUT, CFF, CFR, and RRF.

NOTE

LPCMP may output a glitch and affect the value of CSR[CFF] and CSR[CFR] at the moment of enabling CMP. In order to ensure correctness, it is recommended to write one to clear (W1C) CSR[CFF] and CSR[CFR] before further configuring CMP.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							COUT	0					RRF	CFF	CFR
W														W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-9 —	Reserved
8 COUT	Analog Comparator Output Returns the current value of the analog comparator output when read. This field resets to 0 and reads as CCR1[COUT_INV] after the analog comparator module disables when CCR0[CMP_EN] = 0. Writing to this field is ignored.
7-3 —	Reserved
2 RRF	Round-Robin Flag Detects when any channel's last comparison result is different from the pre-set value in Trigger mode. Write 1 to clear this field. This field clears when CCR0[CMP_EN] or RRCR0[RR_EN] is not 1. 0b - Not detected 1b - Detected
1 CFF	Analog Comparator Flag Falling Detects when a falling edge on COUT occurs. Write 1 to clear this field when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag automatically clears after DMA is done. This field clears when CCR0[CMP_EN] is not 1. 0b - Not detected 1b - Detected
0 CFR	Analog Comparator Flag Rising Detects when a rising edge on COUT occurs. Write 1 to clear this field when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag automatically clears after DMA is done. This field clears when CCR0[CMP_EN] is not 1. 0b - Not detected 1b - Detected

44.7.10 Round Robin Control Register 0 (RRCR0)

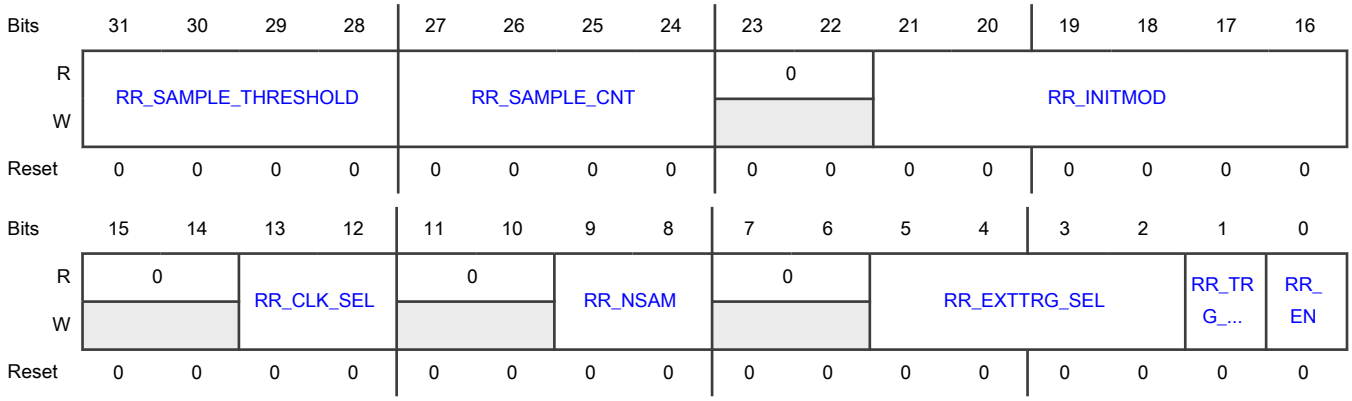
Offset

Register	Offset
RRCR0	24h

Function

Contains configuration options for the round-robin operation, such as enabling it and specifying the initialization delay.

Diagram



Fields

Field	Function
31-28 RR_SAMPLE_THRESHOLD	Sample Time Threshold Specifies that for one channel, when (RR_SAMPLE_THRESHOLD+1) sample results are "1", the final result is "1"; otherwise the final result is "0". **NOTE** This field must not be larger than RRCR0[RR_SAMPLE_CNT]. 0000b - At least 1 sampled "1", the final result is "1" 0001b - At least 2 sampled "1", the final result is "1" 0010b - At least 3 sampled "1", the final result is "1" 0011b - At least 4 sampled "1", the final result is "1" 0100b - At least 5 sampled "1", the final result is "1" 0101b - At least 6 sampled "1", the final result is "1" 0110b - At least 7 sampled "1", the final result is "1" 0111b - At least 8 sampled "1", the final result is "1" 1000b - At least 9 sampled "1", the final result is "1" 1001b - At least 10 sampled "1", the final result is "1" 1010b - At least 11 sampled "1", the final result is "1" 1011b - At least 12 sampled "1", the final result is "1" 1100b - At least 13 sampled "1", the final result is "1" 1101b - At least 14 sampled "1", the final result is "1" 1110b - At least 15 sampled "1", the final result is "1" 1111b - At least 16 sampled "1", the final result is "1"
27-24	Number of Sample for One Channel Specifies the number of samples for one channel.

Table continues on the next page...

Table continued from the previous page...

Field	Function
RR_SAMPLE_CNT	0000b - 1 samples 0001b - 2 samples 0010b - 3 samples 0011b - 4 samples 0100b - 5 samples 0101b - 6 samples 0110b - 7 samples 0111b - 8 samples 1000b - 9 samples 1001b - 10 samples 1010b - 11 samples 1011b - 12 samples 1100b - 13 samples 1101b - 14 samples 1110b - 15 samples 1111b - 16 samples
23-22 —	Reserved
21-16 RR_INITMOD	Initialization Delay Modulus Specifies the number of round-robin clock cycles that determines the comparator and DAC initialization delay specified in the chip datasheet. Calculate the initialization delay as RR_INITMOD * (round-robin clock period). For example, if the initialization delay is 80us and the round-robin clock is 100kHz, program RR_INITMOD to be 80us/10us = 8. 00_0000b - 63 cycles (same as 111111b) 00_0001b-11_1111b - 1 to 63 cycles
15-14 —	Reserved
13-12 RR_CLK_SEL	Round Robin Clock Source Select Selects which clock source is used for the Round Robin clock. See the chip-specific LPCMP information for more on the Round Robin clock source information. 00b - Select Round Robin clock Source 0 01b - Select Round Robin clock Source 1

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>10b - Select Round Robin clock Source 2
11b - Select Round Robin clock Source 3</td></tr>
<tr><td>11-10
—</td><td>Reserved</td></tr>
<tr><td>9-8
RR_NSAM</td><td>Number of Sample Clocks
Specifies the number of the round-robin clock cycles to wait after scanning the active channel before sampling the channel's comparison result. After the next cycle of the round-robin clock, the sampling takes place RR_NSAM clocks later.
00b - 0 clock
01b - 1 clock
10b - 2 clocks
11b - 3 clocks</td></tr>
<tr><td>7-6
—</td><td>Reserved</td></tr>
<tr><td>5-2
RR_EXTTRG_SEL</td><td>External Trigger Source Select
Selects the external trigger from 16 trigger sources. See the chip-specific LPCMP information for more on the trigger source information.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>CMP0</td><td>—</td><td>RRCR0</td></tr>
<tr><td>CMP1</td><td>—</td><td>RRCR0</td></tr>
<tr><td>CMP2</td><td>RRCR0</td><td>—</td></tr>
</table>
0000b - Select external trigger source 0
0001b - Select external trigger source 1
0010b - Select external trigger source 2
0011b - Select external trigger source 3
0100b - Select external trigger source 4
0101b - Select external trigger source 5</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0110b - Select external trigger source 6 0111b - Select external trigger source 7 1000b - Select external trigger source 8 1001b - Select external trigger source 9 1010b - Select external trigger source 10 1011b - Select external trigger source 11 1100b - Select external trigger source 12 1101b - Select external trigger source 13 1110b - Select external trigger source 14 1111b - Select external trigger source 15
1 RR_TRG_SEL	Round-Robin Trigger Select Selects the internal trigger or external trigger as the trigger source. 0b - External trigger 1b - Internal trigger
0 RR_EN	Round-Robin Enable Enables the round-robin operation. 0b - Disable 1b - Enable

44.7.11 Round Robin Control Register 1 (RRCR1)

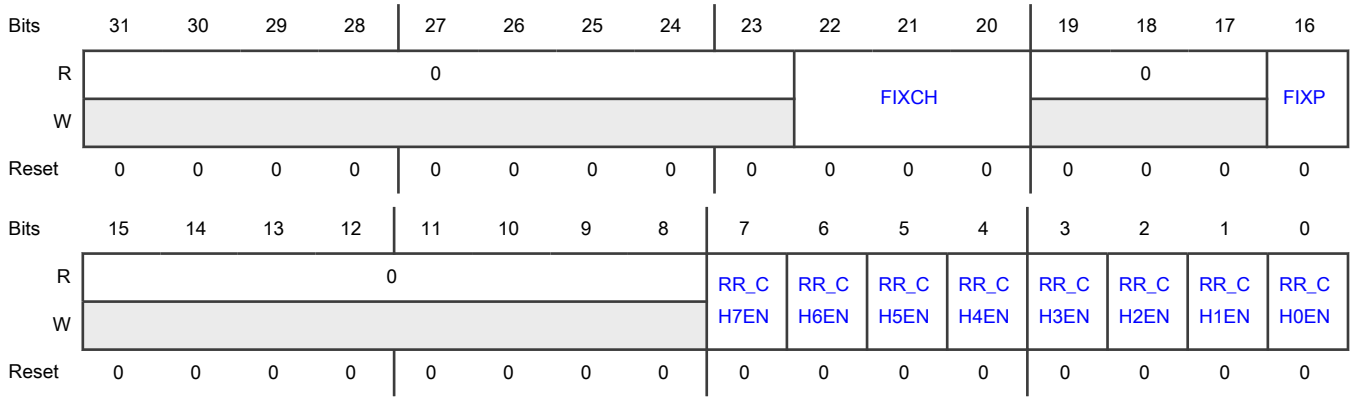
Offset

Register	Offset
RRCR1	28h

Function

Contains configuration options for the round-robin operation, such as enabling individual channels to participate.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-20 FIXCH	Fixed Channel Select Selects which channel in the mux port to fix for a given round-robin trigger mode application. 000b - Channel 0 001b - Channel 1 010b - Channel 2 011b - Channel 3 100b - Channel 4 101b - Channel 5 110b - Channel 6 111b - Channel 7
19-17 —	Reserved
16 FIXP	Fixed Port Fixes an analog mux port (plus or minus) for round-robin trigger mode. The inputs to the non-fixed port sweep during each round. 0b - Fix the plus port. Sweep only the inputs to the minus port. 1b - Fix the minus port. Sweep only the inputs to the plus port.
15-8 —	Reserved
7-0	Channel n Input Enable in Trigger Mode

Table continues on the next page...

Table continued from the previous page...

Field	Function
RR_CHnEN	Enables channel n of the non-fixed mux port to check its voltage value when in Trigger mode. 0b - Disable 1b - Enable

44.7.12 Round Robin Control and Status (RRCSR)

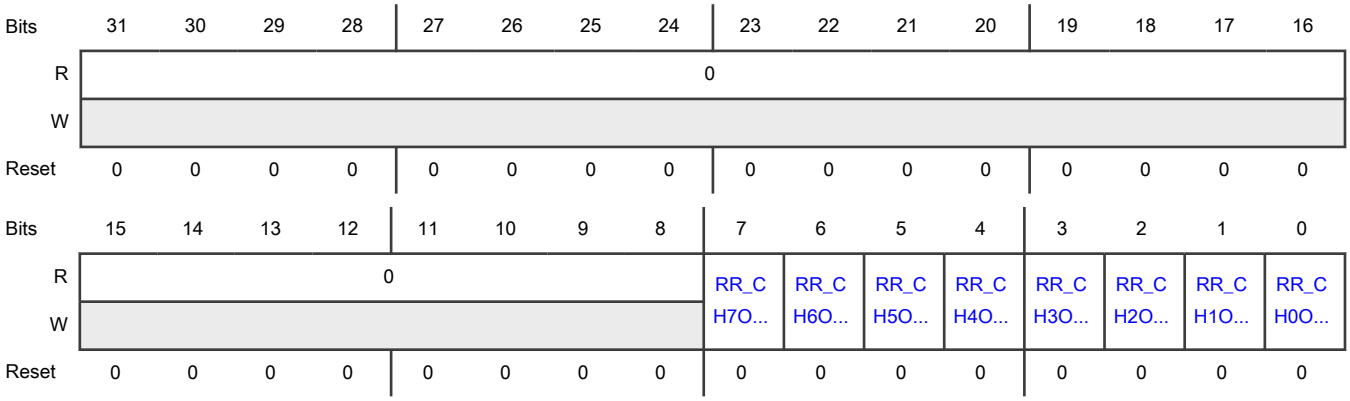
Offset

Register	Offset
RRCSR	2Ch

Function

Contains the latest comparison results of the individual channels with the fixed mux port. It also allows you to define the pre-set state for each channel.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 RR_CHnOUT	Comparison Result for Channel n Returns the latest comparison result for channel n when read and defines the pre-set state for channel n when written to.

44.7.13 Round Robin Status (RRSR)

Offset

Register	Offset
RRSR	30h

Function

Contains individual channel flags that indicates when a channel's last comparison result is different from its pre-set value.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								RR_CH7F	RR_CH6F	RR_CH5F	RR_CH4F	RR_CH3F	RR_CH2F	RR_CH1F	RR_CH0F
W									W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 RR_CHnF	Channel n Input Changed Flag Indicates when the corresponding channel's last comparison result is different from its pre-set value. **NOTE** To clear a flag, write a 1 to it. 0b - No different 1b - Different

44.7.14 Round Robin Control Register 2 (RRCR2)

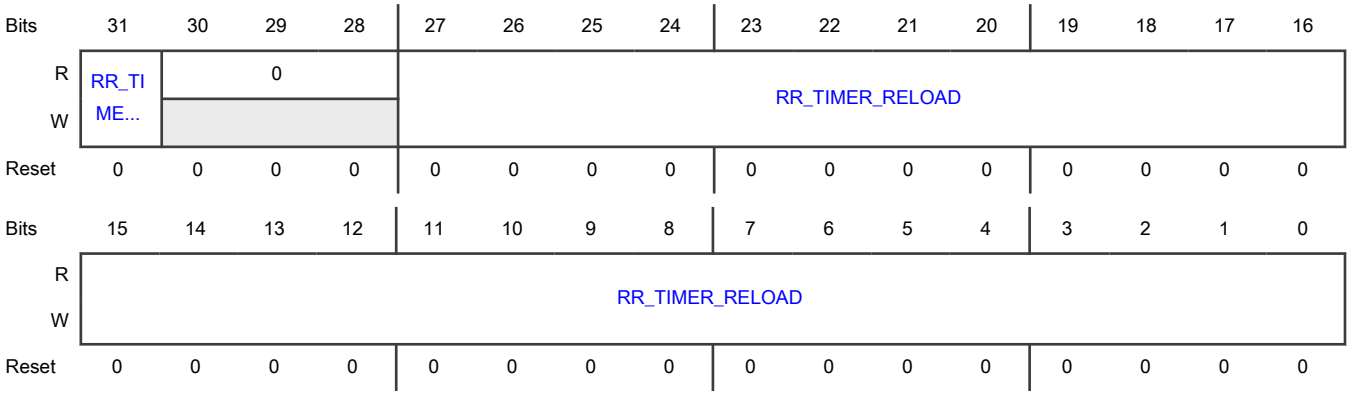
Offset

Register	Offset
RRCR2	38h

Function

Provides the controls for the round-robin internal trigger generation.

Diagram



Fields

Field	Function
31 RR_TIMER_EN	Round-Robin Internal Timer Enable Enables round-robin internal timer. 0b - Disables 1b - Enables
30-28 —	Reserved
27-0 RR_TIMER_RELOAD	Number of Sample Clocks Establishes the repetitive count rate for the round-robin internal timer. Each time the timer counts zero it is reloaded with this value. The round-robin trigger signal generates at a rate of (RR_TIMER_RELOAD + 1) times the round-robin clock period.

Chapter 45
Voltage Reference (VREF)

45.1 Chip-specific VREF information

Table 382. Reference links to related information

Topic	Related module	Reference
Full description	VREF	VREF
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

45.1.1 Module instances

This device has one instance of the VREF module, VREF0.

45.1.2 VREF chip-specific initialization

- Enable the clock to the VREF (AHBCLKCTRL3[VREF] = 1) in SYSCON registers. This enables the register interface and the peripheral function clock.
- Power up the VREF in SPC.ACTIVE_CFG1 register. See SPC.ACTIVE_CFG1[SOC_CNTRL].
- Configure the VREF as described in section Initialization.
  - UTRIM data is automatically loaded with a factory-trimmed after power up. It can also be updated by writing the UTRIM register.

45.1.3 Enabling VREF

To enable reference current, set CSR[LPBGEN] to 1. To enable 1v buffered reference voltage, set CSR[LPBGEN] and CSR[LPBG_BUF_EN].

To use VREF_OUT, set the following bits: CSR[LPBGEN], CSR[HCBGEN], CSR[CHOPEN], CSR[ICOMPEN], CSR[REGEN], CSR[BUF21EN].

45.2 Overview

VREF provides a buffered reference voltage that you can set to levels ranging from 1.0 V to 2.1 V for use as an external reference. When VREF is enabled, the reference voltage, VREF_OUT, is connected to a dedicated output pin. In addition, the buffered reference is available internally for use with on-chip peripherals.

You can trim the VREF voltage output in fine or coarse voltage steps:

- For fine-tuning, use UTRIM[VREFTRIM] to trim the output with (0.05% x VREF_OUT) resolution, where VREF_OUT is the value of the voltage reference output (measured in volts) after using UTRIM[TRIM2V1] to trim the output. For example, after using UTRIM[TRIM2V1] to trim the output VREF_OUT = 1 V, the fine step resolution is : (0.05% x 1) = 0.5 mV.
- For coarse-tuning, use UTRIM[TRIM2V1] to trim the output with 100 mV resolution. For details about possible configurations of the voltage reference, see Table 383

45.2.1 Features

- Programmable User Trim (UTRIM) with 0.5 mV steps in fine-tuning and 100 mV in coarse-tuning. When reset, User Trim (UTRIM) is automatically loaded with a factory-trimmed value.

- Programmable Buffer mode selection:
  - Off
  - Bandgap enabled or standby (output buffer disabled)
  - Low-power (output buffer enabled)
  - High-power (output buffer enabled)
- Low-power bandgap and high-accurate bandgap selection
- Support for a dedicated output pin, VREF_OUT

45.2.2 Block diagram

Figure 206 shows the VREF block diagram.

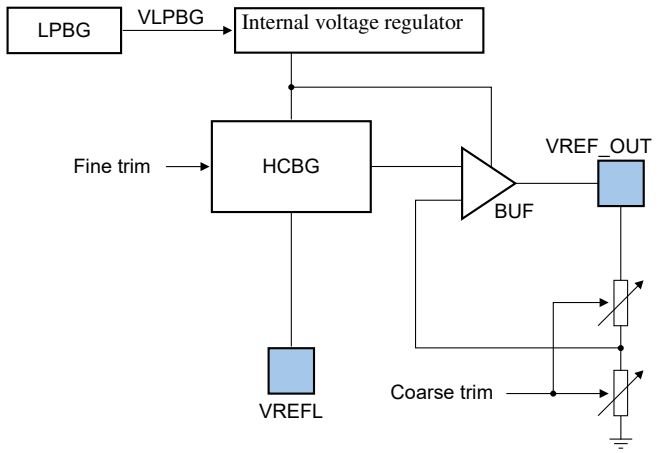


Figure 206. VREF block diagram

45.3 Functional description

VREF is a bandgap buffer system that uses unity gain amplifiers. Both internal and external peripherals use the VREF_OUT signal in Low- and High-Power Buffer modes.

NOTE

When using VREF, a 220 nF capacitor must always be connected between VREF_OUT and Analog Domain Power Supply Ground.

If CSR[HCBGEN] = 1, the voltage reference is enabled, you must set different modes by using CSR[BUF21EN] and CSR[HI_PWR_LV].

The following table shows possible configurations of the voltage reference.

Table 383. VREF_OUT values and related settings

UTRIM[TRIM2V1]	VREF_OUT (V)
0000	1.0
0001	1.1
0010	1.2

Table continues on the next page...

Table 383. VREF_OUT values and related settings (continued)

UTRIM[TRIM2V1]	VREF_OUT (V)
0011	1.3
0100	1.4
0101	1.5
0110	1.6
0111	1.7
1000	1.8
1001	1.9
1010	2.0
1011	2.1
11XX	1.0

45.3.1 Voltage reference disabled

When CSR[LPBGEN] = 0, CSR[HCBGEN] = 0, and CSR[BUF21EN] = 0, the voltage reference is disabled and the VREF bandgap and output buffers are disabled. VREF is in Off mode.

45.3.2 Voltage reference enabled

Enabling voltage reference results into the following main high-level configuration options:

- Write 1 to CSR[LPBGEN]; otherwise, ADC sends a signal to enable the low-power bandgap (LPBG) supply current bias for ADC use.
- Write 1 to CSR[LPBGEN] and CSR[HCBGEN]. This configuration enables the VREF supply voltage on VREF_OUT.

45.3.2.1 CSR[BUF21EN] = 0, CSR[HI_PWR_LV] = X

In Standby mode, the internal VREF bandgap is enabled to generate an accurate 1.0 V output that you can trim by using UTRIM[VREFTRIM]. The bandgap requires time for startup and stabilization. Monitor CSR[VREFST] to determine whether the stabilization and startup are complete when the chop oscillator is not enabled. When chop oscillator is used, the internal bandgap reference voltage settles within the chop oscillator startup time, Tchop_osc_stup.

The output buffer is disabled in this mode, and there is no buffered voltage output. VREF is in Standby mode. If you select this mode first and Low-Power or High-Power Buffer mode is subsequently enabled, there is a delay before the buffer output settles to its final value. This is the buffer startup delay (Tstup), as specified in the appropriate chip data sheet.

45.3.2.2 CSR[BUF21EN] = 1, CSR[HI_PWR_LV] = 1

In Low-Power mode, the internal VREF bandgap is on, and the High-Power Buffer mode is enabled to generate a buffered voltage to VREF_OUT. The available voltage levels range from 1.0 to 2.1 V and can be adjusted in 0.1 V steps using UTRIM[TRIM2V1]. VREF bandgap can also be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from Standby mode (CSR[BUF21EN] = 0), there is a delay before the buffer output settles to its final value. This is the buffer startup delay (Tstup), as specified in the appropriate device data sheet.

If this mode is entered when VREF is enabled, you must wait longer than Tstup or until CSR[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, wait for the time specified by Tchop_osc_stup (chop oscillator startup time) to ensure that the VREF output has stabilized.

NOTE

In this mode, a 220-nF capacitor is required to connect between the VREF_OUT pin and Analog Domain Power Supply Ground.

45.3.2.3 CSR[BUF21EN] = 1, CSR[HI_PWR_LV] = 0

In High-Power mode, the internal VREF bandgap is on, and the low-power buffer is enabled to generate a buffered voltage to VREF_OUT. The available voltage levels range from 1.0 to 2.1 V and you can adjust them in 0.1 V steps using UTRIM[TRIM2V1]. VREF bandgap can be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from Standby mode (CSR[BUF21EN] = 0), there is a delay before the buffer output settles to its final value. This is the buffer startup delay (Tstup), as specified in the appropriate chip data sheet.

If this mode is entered when the VREF module is enabled, you must wait longer than Tstup or until CSR[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, wait for the time specified by Tchop_osc_stup (chop oscillator startup time) to ensure that the VREF output has stabilized.

NOTE

In this mode, a 220-nF capacitor connects the VREF_OUT pin and Analog Domain Power Supply Ground.

45.3.3 Internal voltage regulator

VREF contains an internal voltage regulator that you can enable to provide additional supply noise rejection. The module also contains an internal chop oscillator to generate the chop clock. When possible, enable the internal regulator and chop oscillator to attain optimum VREF performance.

When using the chop function, you must also enable the internal voltage regulator. Perform the following procedure when enabling the internal regulator:

1. Enable the internal regulator by writing 1 to CSR[REGEN].
2. Enable the chop oscillator by writing 1 to CSR[CHOPEN].
3. Write 1 to CSR[LPBGEN] and CSR[HCBGEN].

45.3.4 Low-Power modes

VREF can run when the chip is in a low-power mode. To use the VREF regulator and/or the chop oscillator in a low-power mode, the system reference voltage (also known as the bandgap voltage reference) must also be enabled in these modes. See the chip-specific VREF information on how to enable the system reference voltage.

Having the VREF regulator enabled increases the current consumption. Depending on the requirements of the system application, consider disabling the VREF regulator to minimize the current consumption in low-power modes. However, not using the VREF regulator reduces the accuracy of the output voltage by as much as several mV.

45.3.5 Clocking

See the chip-specific VREF information for the required clocking.

45.3.6 Reset

System reset, VREF convert to initial state with buffer cleared and waiting to load new register.

45.3.7 Interrupts

This module has no interrupts.

45.4 External signals

Table 384. External signals

Signal	Description	I/O
VREF_OUT	Internally generated voltage reference output. When the VREF output buffer is disabled, the VREF_OUT signal state is high-impedance.	O

45.5 Initialization

VREF_OUT requires time for startup and stabilization. After writing 1 to CSR[HCBGEN], monitor CSR[VREFST] to determine whether the stabilization and startup are complete when the chop oscillator is not enabled. If the chop oscillator is enabled, Tchop_osc_stup defines the settling time of the internal bandgap reference. You must wait for this time (Tchop_osc_stup) after the internal bandgap is enabled to ensure that the VREF internal reference voltage is stabilized.

After VREF_OUT is enabled and stabilized, changing the value of CSR[HI_PWR_LV] temporarily destabilizes the output voltage at the VREF_OUT pin, even though CSR[VREFST] remains 1. The time it takes for VREF_OUT to settle again is the same as the buffer startup delay (Tstup), as specified in the appropriate device data sheet. Also, applying a step change of the load current to the VREF_OUT pin incurs some time delay to resettle.

To enable the internal 1.75 V VREF regulator, see Internal voltage regulator for the required sequence. When the internal regulator is disabled, the VREF_OUT voltage is more sensitive to supply voltage variation. Use the internal regulator to achieve optimum VREF_OUT performance.

NOTE

You must write 1 to CSR[CHOPEN], CSR[REGEN], and CSR[ICOMPEN] to achieve the performance stated in the device data sheet.

45.6 Register descriptions

45.6.1 VREF register descriptions

45.6.1.1 VREF memory map

VREF0 base address: 4011_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0100_0000h
8h	Control and Status (CSR)	32	RW	0000_0000h
10h	User Trim (UTRIM)	32	RW	See section

45.6.1.2 Version ID (VERID)

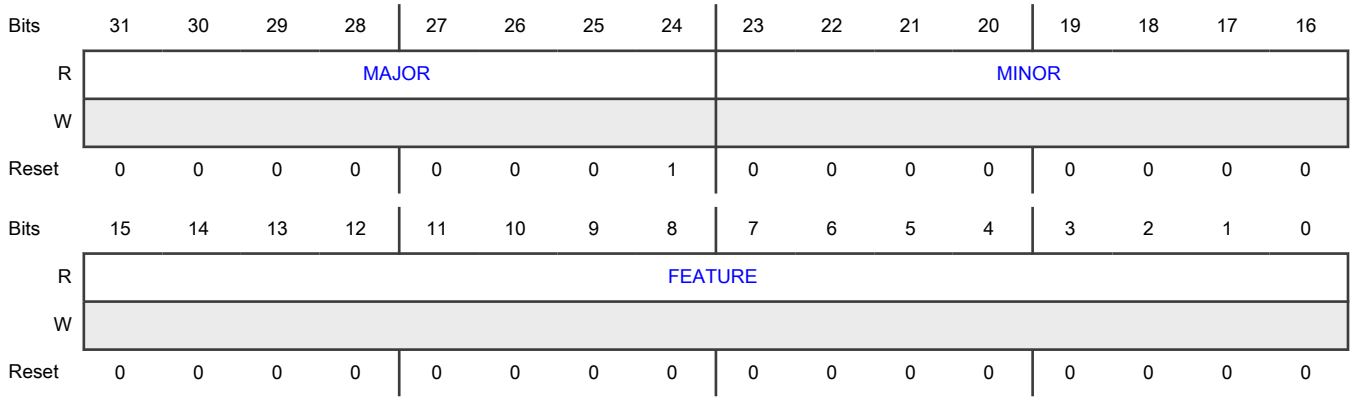
Offset

Register	Offset
VERID	0h

Function

Contains version numbers for the module design and feature set.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15-0 FEATURE	Feature Specification Number Returns the feature set number.

45.6.1.3 Control and Status (CSR)

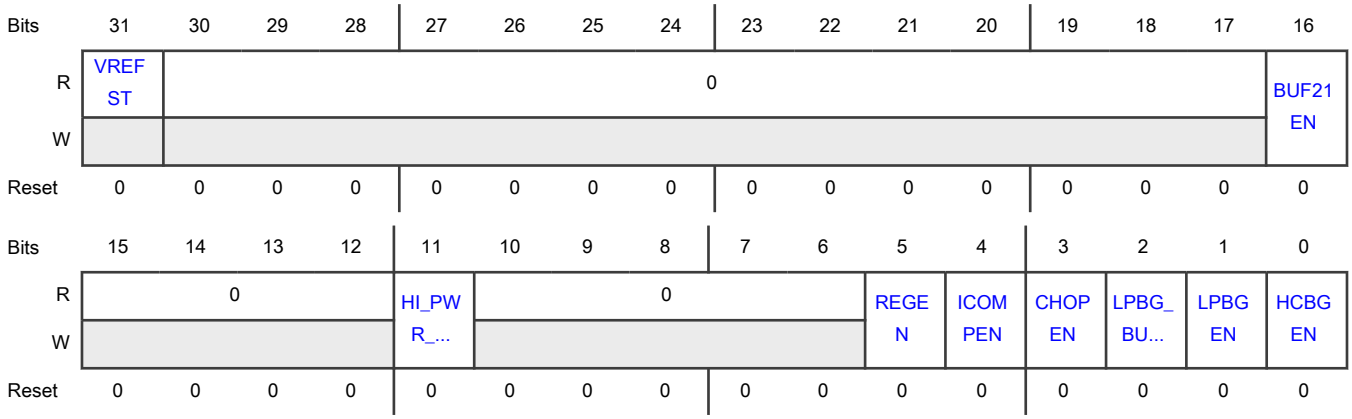
Offset

Register	Offset
CSR	8h

Function

Contains the VREF control and status fields.

Diagram



Fields

Field	Function
31 VREFST	Internal HC Voltage Reference Stable Indicates whether the bandgap reference within VREF has completed its startup and stabilization. **NOTE** This field is valid only when you do not use the chop oscillator. 0b - Disabled and unstable 1b - Stable
30-17 —	Reserved
16 BUF21EN	Internal Buffer21 Enable Enables the Buffer21 programmable output voltage from 1.0 to 2.1 V. 0b - Disables 1b - Enables
15-12 —	Reserved
11 HI_PWR_LV	High-Power Level Controls the power mode of Buffer21. 0b - Low-power 1b - High-power
10-6 —	Reserved
5	Regulator Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
REGEN	Enables the internal 1.75 V regulator to produce a constant internal voltage supply to reduce the sensitivity to external supply noise and variation. See Internal voltage regulator for the required sequence to enable the internal regulator. To keep the regulator enabled in low-power modes, see the chip-specific VREF information for details. **NOTE** You must write 1 to this field to achieve the performance stated in the data sheet. 0b - Disables 1b - Enables
4 ICOMPEN	Current Compensation Enable Enables second-order curvature compensation. **NOTE** You must write 1 to this field to achieve the performance stated in the data sheet. 0b - Disables 1b - Enables
3 CHOPEN	Chop Oscillator Enable Enables the chop oscillator. When set, the internal chopping operation is enabled and the internal analog offset is minimized. When using the internal voltage regulator CSR[REGEN] = 1, you must also enable the chop oscillator. If the chop oscillator is used in low-power modes, you must also enable the HC bandgap voltage reference. See the chip-specific VREF information for details. 0b - Disables 1b - Enables
2 LPBG_BUF_EN	Low-Power Bandgap Buffer Enable Enables the low-power buffer for the low-power bandgap with latch function enabled. **NOTE** You can enable this field only when CSR[LPBGEN] = 1. 0b - Disables 1b - Enables
1 LPBGEN	Low-Power Bandgap Enable Enables the low-power bandgap. 0b - Disables 1b - Enables
0	HC Bandgap Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
HCBGEN	Enables the HC bandgap. **NOTE** You can enable this field only when CSR[LPBGEN] = 1. 0b - Disables 1b - Enables

45.6.1.4 User Trim (UTRIM)

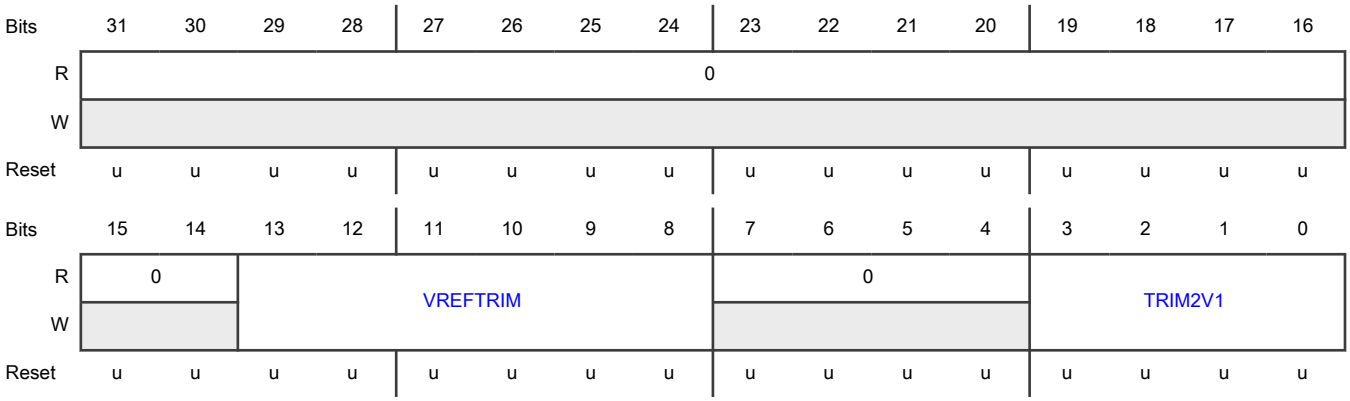
Offset

Register	Offset
UTRIM	10h

Function

Contains the trim data. You can read and write to this register in User mode. After reset, UTRIM is automatically loaded with a factory-trimmed value.

Diagram



Fields

Field	Function
31-14 —	Reserved
13-8 VREFTRIM	VREF Trim

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This is an unsigned number that adjusts the VREF output by (0.5 x VREF_OUT_Nom) mV, where VREF_OUT_Nom is the target output determined by the TRIM2V1 setting. For example, if TRIM2V1 is set to output 1.0 V, VREFTRIM adjusts the output in 0.5 mV steps.
7-4 —	Reserved
3-0 TRIM2V1	VREF 2.1 V Trim Changes the resulting VREF output by approximately ± 100 mV for each step. For details about possible configurations of the voltage reference, see Table 383

Chapter 46
Operational Amplifier (OPAMP)

46.1 Chip-specific OPAMP information

Table 385. Reference links to related information

Topic	Related module	Reference
Full description	OPAMP	OPAMP
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See OPAMPn_TRIG registers in INPUTMUX

46.1.1 Module instances

This device contains three instances of the OPAMP module, OPAMP0, OPAMP1, and OPAMP2.

46.1.2 Positive voltage references

The positive voltage reference can be selected using the OPAMP_CTR[PREF] register. The PREF inputs correspond to following signals.

Table 386. PREF inputs

Register bit field value	PREF input[1]	Corresponds to
00b	Input 0	DACx_OUT • DAC0_OUT for OPAMP0 • DAC1_OUT for OPAMP1 • DAC2_OUT for OPAMP2
01b	Input 1	VDD_ANA/2
10b	Input 2	VREFO
11b	Input 3	520 mv. VREFH provides VREF1V, which is divided inside OPAMP to generate 0.52 V.

1. The voltage range is between 0 - VDD_ANA-0.8.

46.1.3 OPAMP_CTR[ADCSW1]

OPAMP_CTR[ADCSW1] can be used to enable negative gain resistor ladder voltage output to ADC and CMP channel. See ADC input connections and CMP input connections for the channel number (OPAMPx_INT).

46.1.4 OPAMP_CTR[ADCSW2]

OPAMP_CTR[ADCSW2] can be used to enable positive gain resistor ladder voltage output to ADC channel. See ADC input connections to check the channel number (OPAMPx_OBS).

46.1.5 OPAMP_CTR[BIASC]

OPAMP_CTR[BIASC] field settings:

- 01b Increase: +12.5%
- 10b decrease: -12.5%
- 11b further decrease: -25%

46.2 Overview

The OPAMP is an advanced amplifier that supports programmable gain (PGA), inverting, non-inverting, buffer mode and standalone mode. Positive and negative inputs of an amplifier connect to internal channels. The module is applicable to the signal-processing stage before ADC.

46.2.1 Block diagram

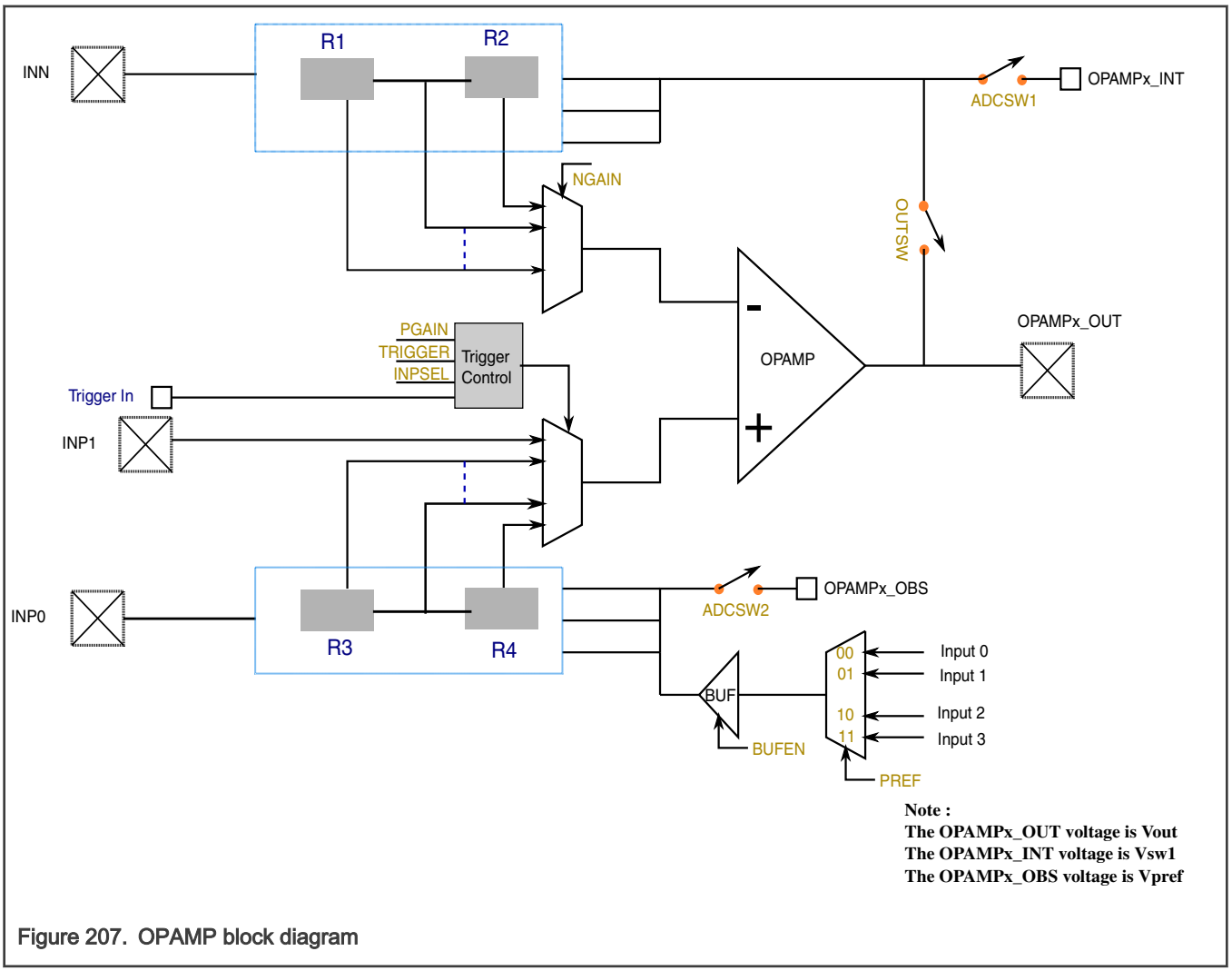


Figure 207. OPAMP block diagram

See chip-specific for supported PREF inputs.

46.2.2 Features

- High performance mode and low power mode

- PGA mode
- Noninverting Amplifier mode
- Inverting Amplifier mode
- Unity gain buffer mode
- Standalone OPAMP mode

46.3 Functional description

This module supports optional gains. See the following table for details where Ngain=R2/R1 and Pgain=R4/R3.

OPAMP_CTR[NGAIN]	Description
000b	Buffer mode
001b	Ngain=1
010b	Ngain=2
011b	Ngain=4
100b	Ngain=8
101b	Ngain=16
110b	Ngain=33
111b	Ngain=64

OPAMP_CTR[PGAIN]	Description
000b	Positive input 1 (INP1)
001b	Pgain=1
010b	Pgain=2
011b	Pgain=4
100b	Pgain=8
101b	Pgain=16
110b	Pgain=33
111b	Pgain=64

NOTE

The actual gain is determined by different usage methods. Some gain calculation examples have been shown in Application Information

46.3.1 Modes

This module supports high performance mode and low power mode selected by configuring OPAMP_CTR[MODE].

The table below shows all supported modes and the related configurations:

Table 387. OPAMP modes configuration

Mode	ADCSW1	OUTSW	ADCSW2	NGAIN	PGAIN	INPSEL	PREF	BUFEN
PGA	ON	ON	ON	x	x	INP0	00b/01b/10b/11b	ON
Noninverting	ON	ON	OFF	x	x	INP0, INP1	-	OFF
Inverting[1]	ON	ON	OFF	x	0	INP1	-	OFF
Standalone	OFF	OFF	OFF	011b	011b	INP0, INP1	-	OFF
Buffer	OFF	OFF	OFF	0	0	INP1	-	OFF

1. Vinp0 must be floating

46.3.1.1 Trigger mode

On the chip, there are up to two pads that can be connected to the OPAMP positive input. In trigger mode, hardware can automatically switch the pad which is connected to OPAMP positive input. It recommends that trigger mode is used when OPAMP is configured to noninverting amplifier. Before enabling trigger mode, the INPSEL bit in the CTR register should be set. Write 0 to BUFEN to disable reference buffer. Configure NGAIN to required gain. After finishing all the configurations, enable the trigger mode. In trigger mode, whenever there is a positive edge of trigger, hardware will switch the pad which is connected to OPAMP positive input. You can read INPF register bit field to monitor which pad is connected to OPAMP positive input in trigger mode.

46.3.1.2 PGA Mode

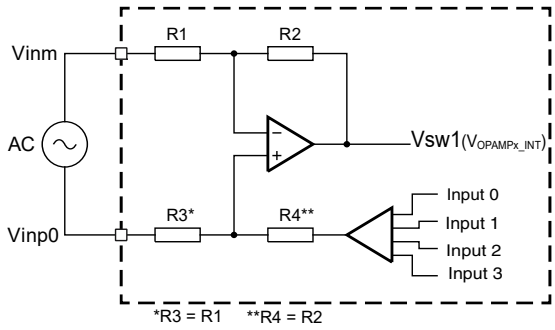


Figure 208. PGA Mode

See chip-specific for supported PREF inputs.

In this mode, configure the registers OPAMP_CTR[NGAIN] and OPAMP_CTR[PGAIN] to select different gains. Enable OPAMP_CTR[BUFEN] and configure the registers OPAMP_CTR[PREF] to select positive reference voltage. Configure OPAMP_CTR[OUTSW] to connect R2 with OPAMP output. Enable OPAMP_CTR[ADCSW1] and OPAMP_CTR[ADCSW2], measure the results from OPAMPx_INT and OPAMPx_OBS by ADC.

46.3.1.3 Noninverting Amplifier mode

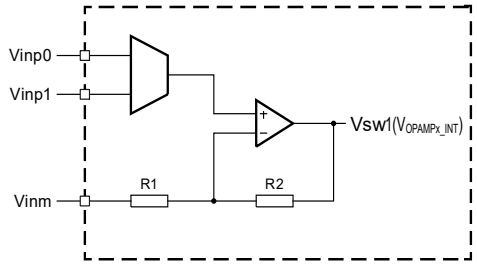


Figure 209. Noninverting Amplifier mode

This mode output a forward gain voltage and the output voltage is connected to the ADC. Configure the registers **OPAMP_CTR[NGAIN]** to select different gains. The positive input can be from Vinp0 or Vinp1. If Vinp1 is used, configure **OPAMP_CTR[PGAIN]** to 0. Configure OPAMP_CTR[BUFEN] to 0 to disable reference buffer. Enable OPAMP_CTR[OUTSW] to connect R2 with OPAMP output (OPAMPx_OUT). Enable OPAMP_CTR[ADCSW1], and measure results from OPAMPx_INT.

46.3.1.4 Inverting Amplifier Mode

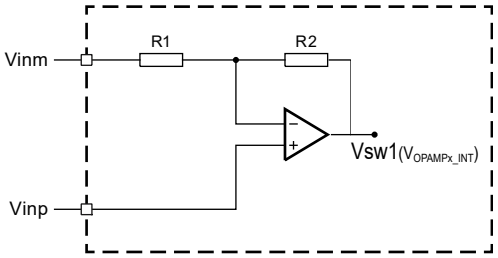


Figure 210. Inverting Amplifier Mode

This mode output a reverse gain voltage and the output voltage is connected to the ADC. Configure the registers **OPAMP_CTR[NGAIN]** to select different gains. The positive input can be either from Vinp1 or reference buffer. Configure **OPAMP_CTR[PGAIN]** to 0 to select Vinp1. Enable reference buffer by **OPAMP_CTR[BUFEN]** to select reference buffer as the source of positive input. Vinp0 must be floating. Enable **OPAMP_CTR[OUTSW]** to connect R2 with OPAMP output (OPAMPx_OUT). Enable **OPAMP_CTR[ADCSW1]**, and measure results from OPAMPx_INT.

46.3.1.5 Standalone OPAMP Mode

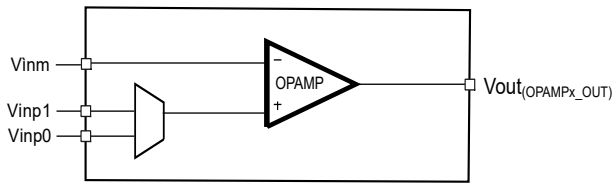


Figure 211. Standalone OPAMP Mode

To use standalone OPAMP mode, disconnect R2 with OPAMP output by disabling **OPAMP_CTR[OUTSW]**. Configure **OPAMP_CTR[NGAIN]=011b** and **OPAMP_CTR[PGAIN]=011b**. Disable reference buffer by **OPAMP_CTR[BUFEN]**. Disconnect ADCSW1 and ADCSW2.

46.3.1.6 Buffer Mode

To select buffer mode, configure the value of registers **OPAMP_CTR[NGAIN]** and **OPAMP_CTR[PGAIN]** to 000b. The output voltage is connected to the **OPAMPx_OUT**. Configure **OPAMP_CTR[BUFEN]** to 0 to disable reference buffer. Enable **OPAMP_CTR[OUTSW]** to connect R2 with OPAMP output. Enable **OPAMP_CTR[ADCSW1]**, and measure results from **OPAMPx_INT**.

46.3.2 Clock

The global clock is used for OPAMP.

46.3.3 Reset

OPAMP is reset by the global chip reset signal.

46.3.4 Interrupts

This module has no interrupts.

46.4 External Signals

Table 388. External Signal Description

Signal	Description	I/O
INN	Analog input channels(Core amplifier negative input channel)	I
INP0	Analog input channels(Core amplifier positive input channel0)	I
INP1	Analog input channels(Core amplifier positive input channel1)	I
V_{OPAMPx_OUT}	Output of the core amplifier	O

46.5 Initialization

This module does not require initialization.

46.6 Application Information

Below is the OPAMP output equation :

$$V_{sw1} = \frac{Ngain+1}{1+\frac{1}{Pgain}} V_{INP0} - Ngain * V_{INN} + \frac{1+Ngain}{1+Pgain} V_{pref}$$

46.6.1 PGA mode Application Information

- Both OUTSW and ADCSW1/2 switches should be on.
- OPAMP_CTR[NGAIN] and OPAMP_CTR[PGAIN] should be set to same value.
- Measurement method:
  - Two steps method
    1. Measure OPAMPx_OBS voltage, defined as Vpref (voltage input selected by PREF bit) first with ADC single end mode, and the measured Vpref will be used in following calculation.

2. Measure OPAMPx_INT voltage, defined as Vsw1 with ADC single end mode real time.

— Differential method

1. Use ADC differential channel measure Vsw1 and Vpref at same time.

- ADCSW1 could be on to fine-tune the results
- Result :

  Recommend Ngain=Pgain,

  — Vsw1 –Vpref = (Vinp0-Vinm) * Ngain

46.6.2 Noninverting mode Application Information

- ADCSW1 should be closed
- Disable the OPAMP_CTR[BUFEN]
- Result

  — Vsw1 + Ngain * Vinm= (1+Ngain) * Vinp

  NXP recommends you to connect Vinm to GND, if you do not need to provide a DC offset, and Vinp is recommended to be connected to Vinp1.

46.6.3 Inverting mode Application Information

- ADCSW1 should be closed
- Disable the OPAMP_CTR[BUFEN]
- Result

  — Vsw1 = (Ngain+1) * Vinp-Ngain * Vinm

- OPAMP_CTR[PGAIN] is recommended to be set to 000b.

  NXP recommends you to connect Vinp to Vinp1. The recommended voltage of Vinp0/1 is half of VDD_ANA. Vinp cannot be 0.

46.6.4 Buffer mode Application Information

- The unity gain buffer mode supports OPAMPx_OUT or OPAMPx_INT as the output, with Vout defined as the voltage.
- Disable the OPAMP_CTR[BUFEN]
- Result :-

  — Vout=Vinp1

  Vinp is recommended to be connected to Vinp1.

46.7 Memory map and register definition

This section includes the OPAMP module memory map and detailed descriptions of all registers. You can select different gains by configuring registers. Registers have options for selecting noninverting and inverting gain application.

46.7.1 OPAMP register descriptions

46.7.1.1 OPAMP memory map

OPAMP0 base address: 4011_0000h

OPAMP1 base address: 4011_3000h

OPAMP2 base address: 4011_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0000_0000h
4h	Parameter (PARAM)	32	R	0000_0001h
8h	OPAMP Control (OPAMP_CTR)	32	RW	0100_0000h

46.7.1.2 Version ID (VERID)

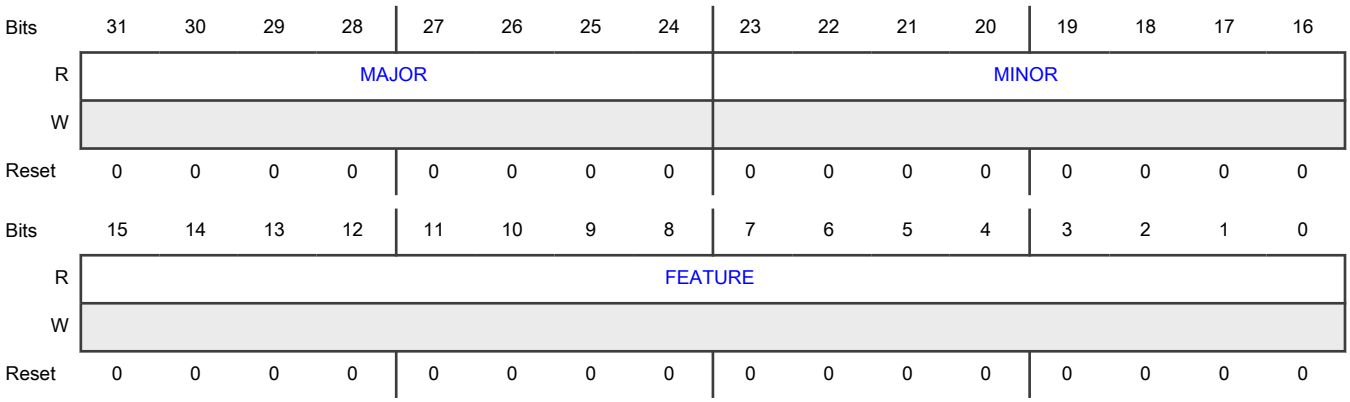
Offset

Register	Offset
VERID	0h

Function

Version ID indicates the version integrated for this instance on the chip. It also indicates inclusion/exclusion of several optional features.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the specification.
15-0 FEATURE	Feature Specification Number Returns the feature set number.

46.7.1.3 Parameter (PARAM)

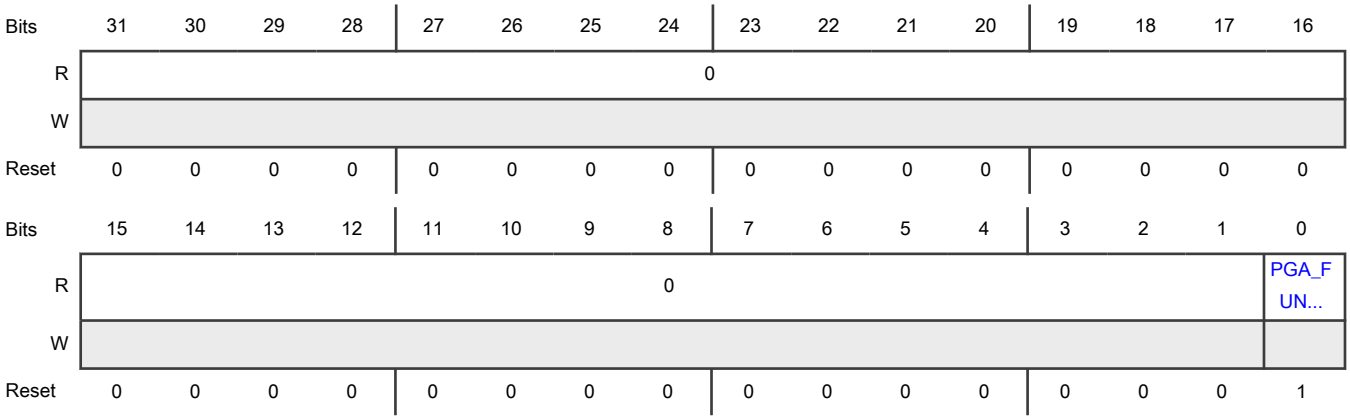
Offset

Register	Offset
PARAM	4h

Function

PARAM indicates optional PGA functions or core amplifier for this instance on the chip.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 PGA_FUNCTION	PGA Function Option Selects different PGA functions. 0b - Core amplifier enabled 1b - PGA function enabled

46.7.1.4 OPAMP Control (OPAMP_CTR)

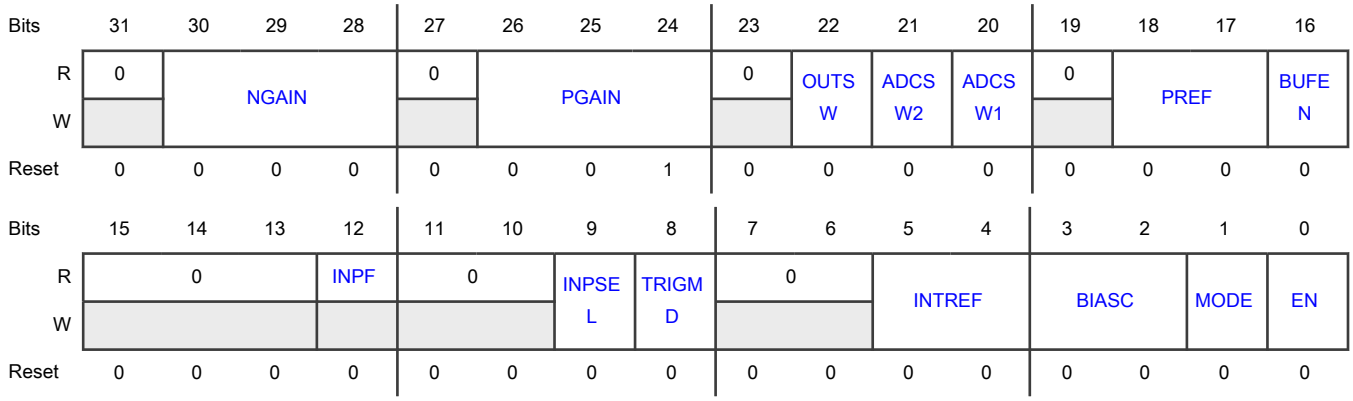
Offset

Register	Offset
OPAMP_CTR	8h

Function

OPAMP_CTR is used to configure the OPAMP module.

Diagram



Fields

Field	Function
31 —	Reserved
30-28 NGAIN	Negative PGA Selection Selects negative programmable gains. 000b - Buffer 001b - Ngain=1 010b - Ngain=2 011b - Ngain=4 100b - Ngain=8 101b - Ngain=16 110b - Ngain=33 111b - Ngain=64
27 —	Reserved
26-24 PGAIN	Positive PGA Selection Selects positive programmable gains. OPAMP cannot write a value to this field when in trigger mode. 000b - Positive input 1 (INP1) 001b - Pgain=1 010b - Pgain=2 011b - Pgain=4 100b - Pgain=8 101b - Pgain=16

Table continues on the next page...

Table continued from the previous page...

Field	Function
	110b - Pgain=33 111b - Pgain=64
23 —	Reserved
22 OUTSW	Output Switch Provides output of OPAMP to the negative gain resistor ladder switch. 0b - OPAMP out to negative gain resistor ladder switch off 1b - OPAMP out to negative gain resistor ladder switch on
21 ADCSW2	Measure Switch 2 Specifies whether the measure positive gain resistor ladder reference voltage switch is on or off for ADC channel. 0b - Measure positive gain resistor ladder reference voltage switch off 1b - Measure positive gain resistor ladder reference voltage switch on
20 ADCSW1	Measure Switch 1 Specifies whether the measure negative gain resistor ladder voltage switch is on or off for ADC channel. 0b - Measure negative gain resistor ladder voltage switch off 1b - Measure negative gain resistor ladder voltage switch on
19 —	Reserved
18-17 PREF	Positive Reference Voltage Selection Selects positive reference voltage. See chip-specific for supported PREF inputs. 00b - Input 0 01b - Input 1 10b - Input 2 11b - Input 3
16 BUFEN	Reference Buffer Enables reference buffer. 0b - Disables 1b - Enables
15-13 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 INPF	Positive Input Connection Status The channel which is connected to OPAMP positive input. In trigger mode, it can use INPF to monitor OPAMP positive input connection. 0b - Positive input 0 (INP0) 1b - Positive input 1 (INP1)
11-10 —	Reserved
9 INPSEL	Positive Input Channel Selection When OPAMP is not in trigger mode, use this field to select positive input channel. Before enabling trigger mode, INPSEL can be used to select the initial connection. After trigger mode is enabled, INPSEL does not change when there are triggers. If PGAIN is configured to 0, and INPSEL is configured to 1, INP1 is connected to OPAMP positive input. OPAMP cannot write a value to this field when in trigger mode. 0b - When OPAMP is not in trigger mode, select positive input 0 (INP0) 1b - When OPAMP is not in trigger mode, select positive input 1 (INP1)
8 TRIGMD	Trigger Mode Enables trigger mode 0b - Disable 1b - Enable
7-6 —	Reserved
5-4 INTREF	Provide OPAMP rail to rail voltage selection 00b - Select OPAMP input rail to rail voltage from 0 to VDD_ANA 01b - Select OPAMP input rail to rail voltage from 0 to VDD_ANA-0.8V 10b - Select OPAMP input rail to rail voltage from 0.8V to VDD_ANA 11b - Not allowed
3-2 BIASC	Bias Current Trim Selection 00b - Default 01b - Increase current 10b - Decrease current 11b - Further decrease current
1 MODE	Mode Selection Selects one mode between High performance mode and Low power mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - High performance mode 1b - Low power mode
0 EN	OPAMP Enable Enables OPAMP. 0b - Disable 1b - Enable

Chapter 47
SINC Filter

47.1 Chip-specific SINC information

Table 389. Reference links to related information

Topic	Related module	Reference
Full description	SINC	SINC
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See SINC_FILTER_CHn registers in INPUTMUX

47.1.1 Module instances

This device has one instance of the SINC Filter module, SINC0.

47.1.2 SINC output signals

The conversion complete signal is routed into SCT, ADC0/1 and FlexIO via INPUTMUX.

The Break signals are routed into PWM0/1 and EVTG via INPUTMUX. Five break signals are connected using OR into one signal into INPUTMUX.

47.1.3 Wake-up

This chip does not support the Wake-up feature.

47.1.4 SINC pin functions

SINC Filter data input and output pin functions are mapped to various SoC pins and are selected by the Pin Function registers (see Signal multiplexing for more details). The table below lists the SINC Filter data input and output pin functions mapped to the SoC-level pins.

NOTE

Not all SoC pins are pinned out on all packages.

Table 390. SINC data input pin functions

SINC Filter input pin	SoC pin	SoC pin
MCLK0	P2_2	P4_12
MCLK1	P2_4	P4_16
MCLK2	P2_6	P4_20
MCLK3	P2_8	P4_0
MCLK4	P2_10	P4_4

Table continues on the next page...

Table 390. SINC data input pin functions (continued)

SINC Filter input pin	SoC pin	SoC pin
MBIT0	P2_3	P4_13
MBIT1	P2_5	P4_17
MBIT2	P2_7	P4_21
MBIT3	P2_9	P4_2
MBIT4	P2_11	P4_5

Table 391. SINC data output pin functions

SINC Filter output pin	SoC pin	SoC pin
MCLK_OUT0	P2_1	P4_15
MCLK_OUT1	P4_19	
MCLK_OUT2	P4_23	

47.1.5 SINC external trigger connections

The SINC Filter external trigger inputs come from various sources within the SoC and are selected by the SINC Filter peripheral input muxes residing at INPUTMUX_BASE address plus offsets 0x520 – 0x530 (see INPUTMUX chapter). The trigger input mux at 0x520 selects the external trigger for channel 0 input source, the trigger input mux at 0x524 selects the external trigger for channel 1 input source, and so on. The table below lists the triggers mapped to the SINC Filter input sources.

Table 392. SINC external trigger connections

Number	Input source
0	PIN_INT0
1	PIN_INT1
2	SCT_OUT4
3	SCT_OUT5
4	SCT_OUT9
5	T0_MAT3
6	T1_MAT3
7	T2_MAT3
8	T3_MAT3
9	T4_MAT3
10	Reserved
11	Reserved
12	GPIOINT_BMATCH
13	ADC0_tcomp0
14	ADC0_tcomp1

Table continues on the next page...

Table 392. SINC external trigger connections (continued)

Number	Input source
15	ADC0_tcomp2
16	ADC0_tcomp3
17	ADC1_tcomp0
18	ADC1_tcomp1
19	ADC1_tcomp2
20	ADC1_tcomp3
21	CMP0_OUT
22	CMP1_OUT
23	CMP2_OUT
24	PWM0_SM0_MUX_TRIG0
25	PWM0_SM0_MUX_TRIG1
26	PWM0_SM1_MUX_TRIG0
27	PWM0_SM1_MUX_TRIG1
28	PWM0_SM2_MUX_TRIG0
29	PWM0_SM2_MUX_TRIG1
30	PWM0_SM3_MUX_TRIG0
31	PWM0_SM3_MUX_TRIG1
32	PWM1_SM0_MUX_TRIG0
33	PWM1_SM0_MUX_TRIG1
34	PWM1_SM1_MUX_TRIG0
35	PWM1_SM1_MUX_TRIG1
36	PWM1_SM2_MUX_TRIG0
37	PWM1_SM2_MUX_TRIG1
38	PWM1_SM3_MUX_TRIG0
39	PWM1_SM3_MUX_TRIG1
40	ENC0_CMP/POS_MATCH
41	ENC1_CMP/POS_MATCH
42	EVTG_OUT0A
43	EVTG_OUT0B
44	EVTG_OUT1A
45	EVTG_OUT1B
46	EVTG_OUT2A

Table continues on the next page...

Table 392. SINC external trigger connections (continued)

Number	Input source
47	EVTG_OUT2B
48	EVTG_OUT3A
49	EVTG_OUT3B
50	LPTMR0
51	LPTMR1
52	FlexIO_CH0
53	FlexIO_CH1
54	FlexIO_CH2
55	FlexIO_CH3
56	WUU

47.2 Overview

SINC converts an external ADC sigma-delta modulator bitstream to a data stream. The converters are based on:

- A maximum of 3-order sinc digital decimation filters with FastSinc support.
- A maximum selectable oversampling ratio (OSR) of 2048.

You can combine SINC functionality with additional software-based filtering.

47.2.1 Block diagram

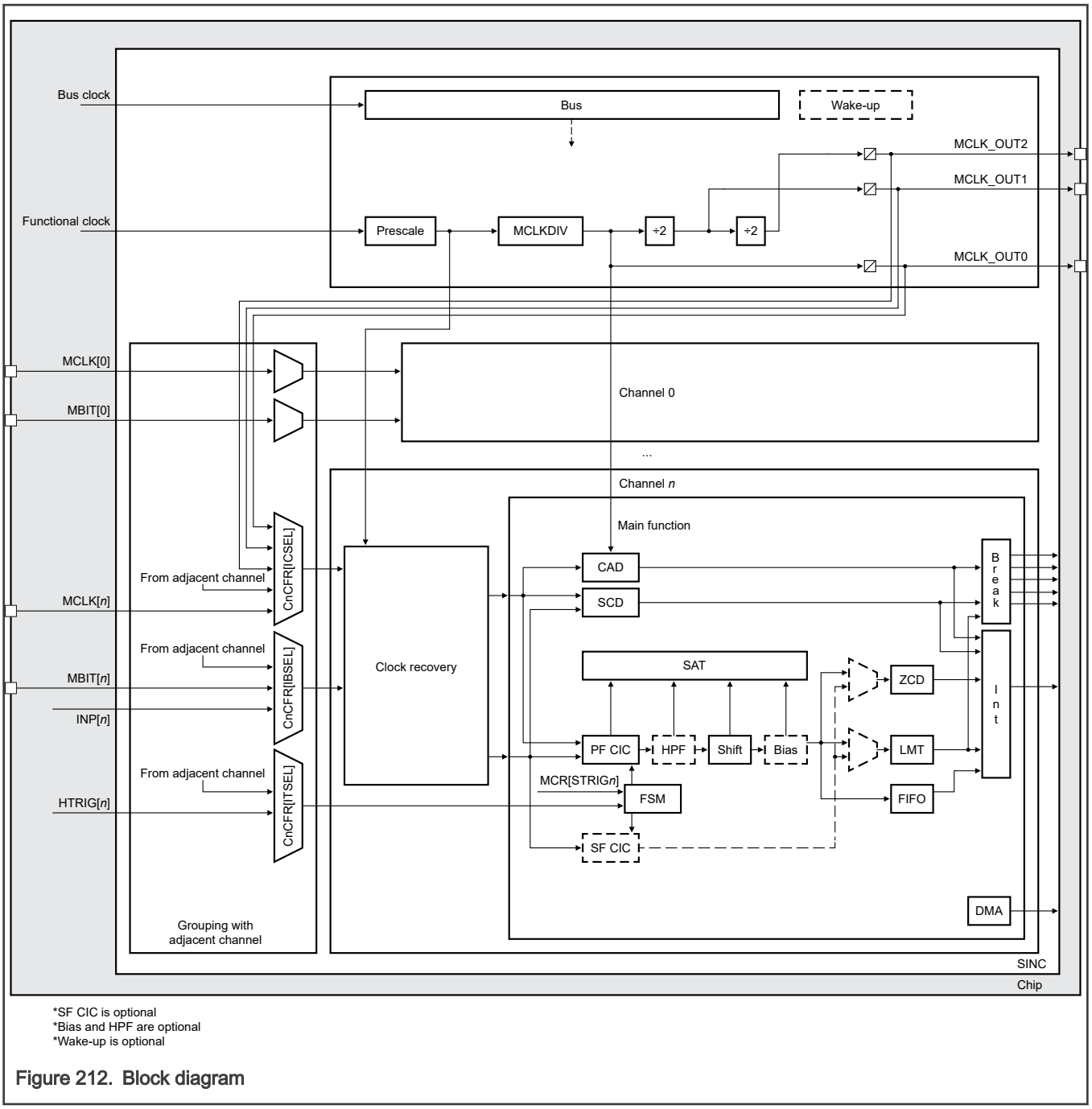


Figure 212. Block diagram

47.2.2 Features

- Filter channels with dedicated modulator clocks, bitstreams, and triggers
- Modulator clock outputs
- One software and one hardware trigger per channel
- One primary filter (PF) for each channel, with a shift and cascaded integrator comb (CIC) filter, for 24-bit precision
- Break signal outputs to turn off the external peripheral when a protection event occurs

- A 8-entry FIFO with watermark support for each channel
- Two DMA requests per channel with various event sources
- Various interrupt sources available for each channel (see Interrupts and breaks)
- Programmable high and low detection limits
- Support for short-circuit detector (SCD), clock-absence detector (CAD), and zero-cross detector (ZCD)
- Programmable shift operations
- Programmable bias (offset) operations
- high-pass filters (HPF) for DC removal
- Support for a Manchester-encoded bitstream

47.3 Functional description

47.3.1 Components

SINC includes the following two main components.

Table 393. Components

Component	Supported functionality
Decimation filter	• A PF per channel • A high-precision 24-bit result with gain scaling • Facility for DC removal using an HPF • Offset bias compensation logic
Digital protection logic	• SCD • Hi-low limit detection (LMT) • CAD • ZCD

47.3.2 Operations

SINC performs operations based on:

- Modulator bitstream sources. See Bitstream source selection for more information.
- Conversion modes. See Modes and behaviors for more information.
- The selected PF. See PF for more information on SINC PF components, and see Calculating ORD and Equation for OSR that define the calculation for ORD and OSR, respectively, of the PF.

47.3.2.1 Calculating ORD

Use the following table to calculate ORD for the PF.

Table 394. Calculating ORD

When C*n*DR[PFORD] is:	ORD is:
0	4
> 0	C*n*DR[PFORD]

47.3.2.2 Equation for OSR

$$OSR = C\mathit{n}DR[PFOSR] + 1$$

Equation 20. Equation for OSR

47.3.2.3 Bitstream source selection

These are SINC's modulator bitstream sources:

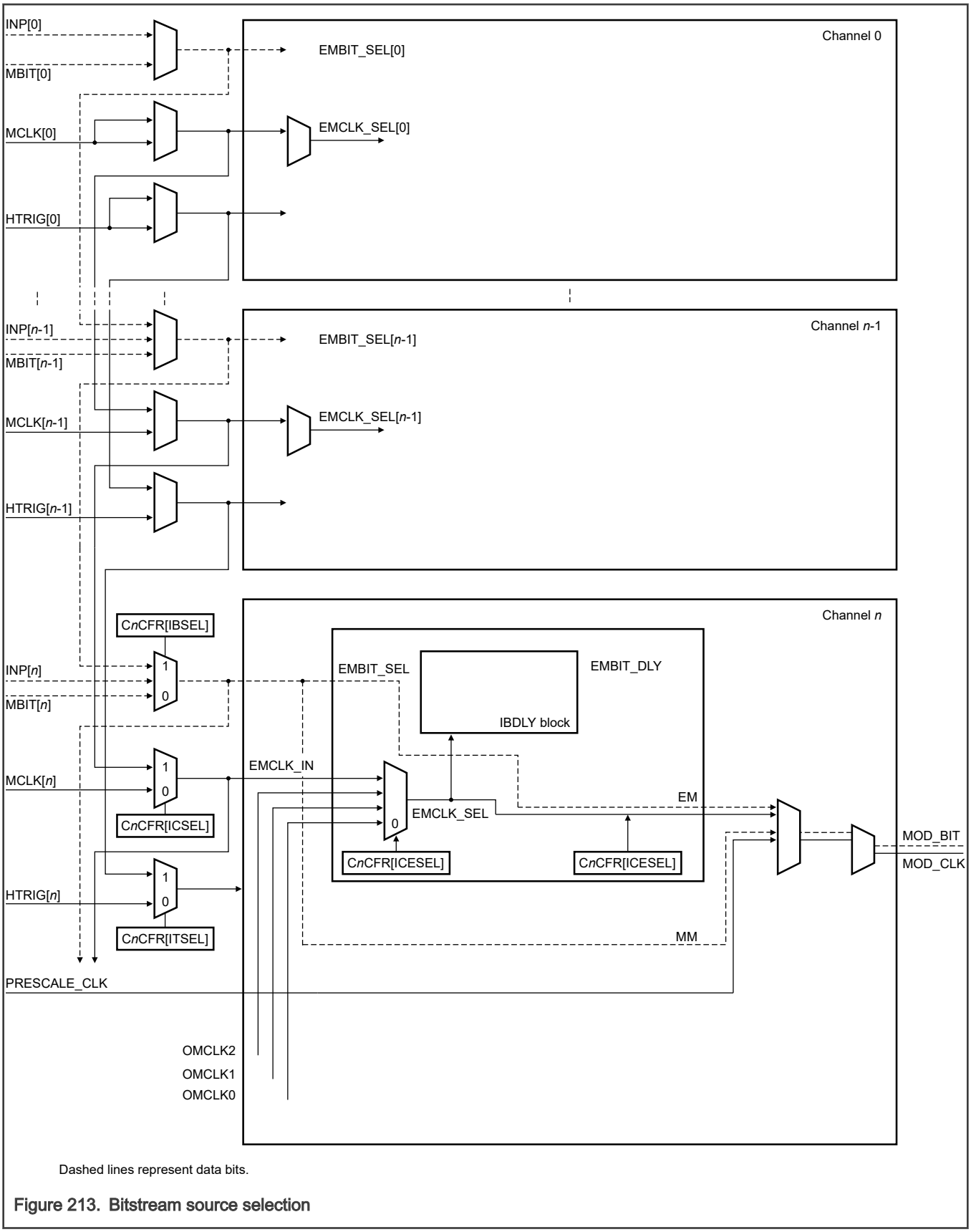
- The MBIT[3:0] signals (see External signals)
- The INP[3:0] signals (see External signals)
- Internal data from registers in Parallel or Serial mode

The following diagram shows MCLK[*n*] and MBIT[*n*] decoder circuits along with these key components:

- External modulator (EM) bitstream (see EM clock input bitstream sampling for EM clock input bitstream sampling)
- Manchester modulator (MM) bitstream (see MM clock input bitstream sampling for MM clock input bitstream sampling)
- Parallel modulator (PM) bitstream (see PM clock input bitstream sampling for PM clock input bitstream sampling)
- Serial modulator (SM) bitstream (see SM clock input bitstream sampling for SM clock input bitstream sampling)

The diagram also illustrates an example of bitstream source selection using:

- An external MCLK[*n*]
- An internal route back clock (OMCLK)
- An internal auto-generated clock for Parallel or Serial mode
- An internal auto-generated clock for Manchester format



Dashed lines represent data bits.

Figure 213. Bitstream source selection

47.3.2.3.1 Modulator bitstream and channel clock selection

Following is the modulator bitstream and clock mux configuration you could perform for each channel.

Table 395. Modulator bitstream and channel clock selection

Use C*n*CFR[IBSEL] to	Use C*n*CFR[ICSEL] to	Use C*n*CFR[ITSEL] to
Select a modulator input bitstream (EMBIT_SEL) for channel *n* that can be either of the following: • A dedicated external modulator bitstream (MBIT[*n*]) if C*n*CFR[IBSEL] = 00b • A dedicated internal modulator bitstream (INP[*n*]) if C*n*CFR[IBSEL] = 01b • A shared group bitstream from an adjacent channel (*n*-1) if C*n*CFR[IBSEL] = 11b	Select a modulator input clock (EMCLK_SEL) for channel *n* that can be either of the following: • A dedicated modulator clock (MCLK[*n*]) if C*n*CFR[ICSEL] = 011b • A shared group clock (MCLK[*n*-1]) from an adjacent channel if C*n*CFR[ICSEL] = 111b • One of the three internal clocks routed to the chip modulator output clock if C*n*CFR[ICSEL] = 000b, 001b, or 010b	Select a modulator input trigger (HTRIG_SEL) for channel *n* that can be either of the following: • A dedicated software trigger (STRIG[*n*]) if C*n*CFR[ITSEL] = 00b • A dedicated hardware trigger (HTRIG[*n*]) if C*n*CFR[ITSEL] = 01b • A shared group hardware trigger (HTRIG[*n*-1]) from an adjacent channel if C*n*CFR[ITSEL] = 11b

47.3.2.3.2 Cross channel modulator bitstreams and clock groups

Following is the modulator bitstream and clock mux configuration you could perform with an adjacent channel.

Table 396. Cross channel modulator bitstreams and clock groups

Write 111b to C*n*CFR[IBSEL] for	Write 111b to C*n*CFR[ICSEL] for	Write 11b to C*n*CFR[ITSEL] for
Combining the adjacent channel modulator bitstreams as groups, which: • Share the same modulator bitstreams and clocks if C*n*CFR[ICSEL] = 111b. • Can be configured to use one channel sample by rising edge and another channel sample by falling edge as a microphone use case. • Can be cascaded.	Combining adjacent channels as groups for the same clock to do either of these: • Reduce the I/O pin number. • Retain channels with the same time sequence. • Cascade, possibly.	Combining adjacent channels as groups for the same hardware trigger to: • Retain channels with the same time sequence. • Share the hardware trigger. • Cascade, possibly.

47.3.2.3.3 Modulator bitstream clock edge selection

Following are the modulator bitstream clock and trigger options you could use for each channel.

Table 397. Modulator bitstream clock edge selection

Use C*n*CFR[ICESEL] to	Use C*n*CFR[ITLVL] to
Select a clock edge and use the selected modulator input clock (EMCLK_SEL) as either of the following:	Select a modulator input trigger event type. The selected modulator input trigger (HTRIG_SEL) can be used as either of the following:

Table continues on the next page...

Table 397. Modulator bitstream clock edge selection

Use CnCFR[ICESEL] to	Use CnCFR[ITLVL] to
• Positive edge (by writing 001b to CnCFR[ICESEL]) • Negative edge (by writing 010b to CnCFR[ICESEL]) • Dual edge (by writing 011b to CnCFR[ICESEL]) • Every other even positive edge (by writing 101b to CnCFR[ICESEL]) • Every other odd negative edge (by writing 110b to CnCFR[ICESEL]) • Every other odd positive edge (by writing 100b to CnCFR[ICESEL]) • Every other even negative edge (by writing 111b to CnCFR[ICESEL])	• A positive edge trigger (by writing 0b to CnCFR[ITLVL]) • A high-level trigger (by writing 1b to CnCFR[ITLVL])

47.3.2.3.4 Modulator bitstream programmed delay

You can write an appropriate value to CnACFR[IBDLY] for the selected modulator bitstream to be delay sampled by the number of prescaled clock cycles. The delay number can be 0 (bypass), and extend to a maximum of 15 clock cycles.

See the following figure for a pictorial representation of the modulator bitstream programmed delay and see Modulator bitstream delay case 0, Modulator bitstream delay case 1, Modulator bitstream delay case 2, Modulator bitstream delay case 3, and Modulator bitstream delay case 4 for delay cases.

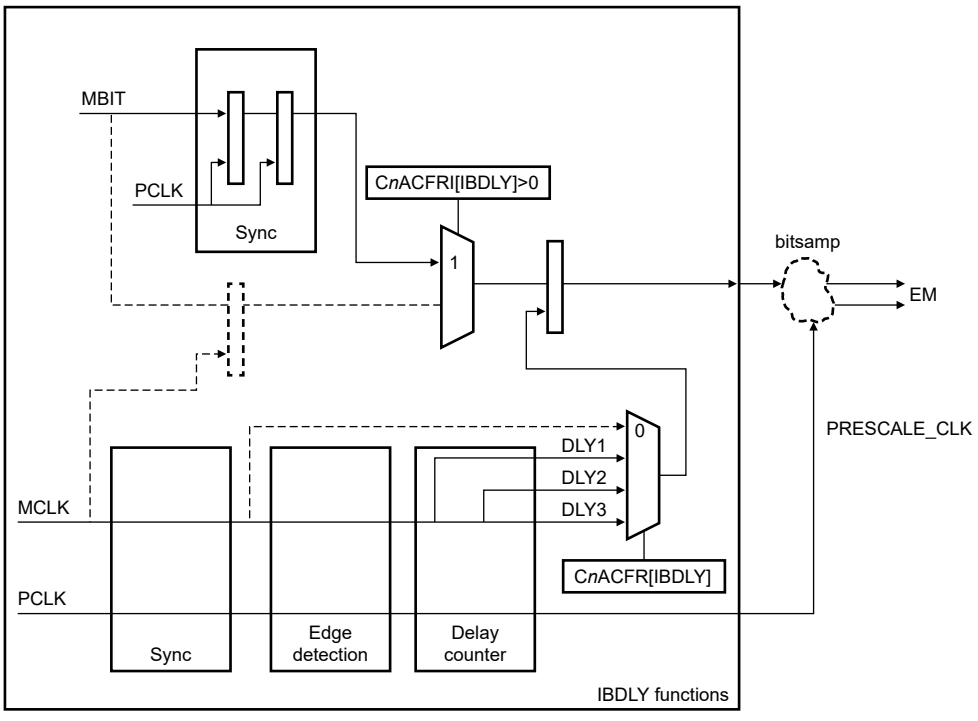


Figure 214. Modulator bitstream programmed delay

47.3.2.3.4.1 Modulator bitstream delay case 0

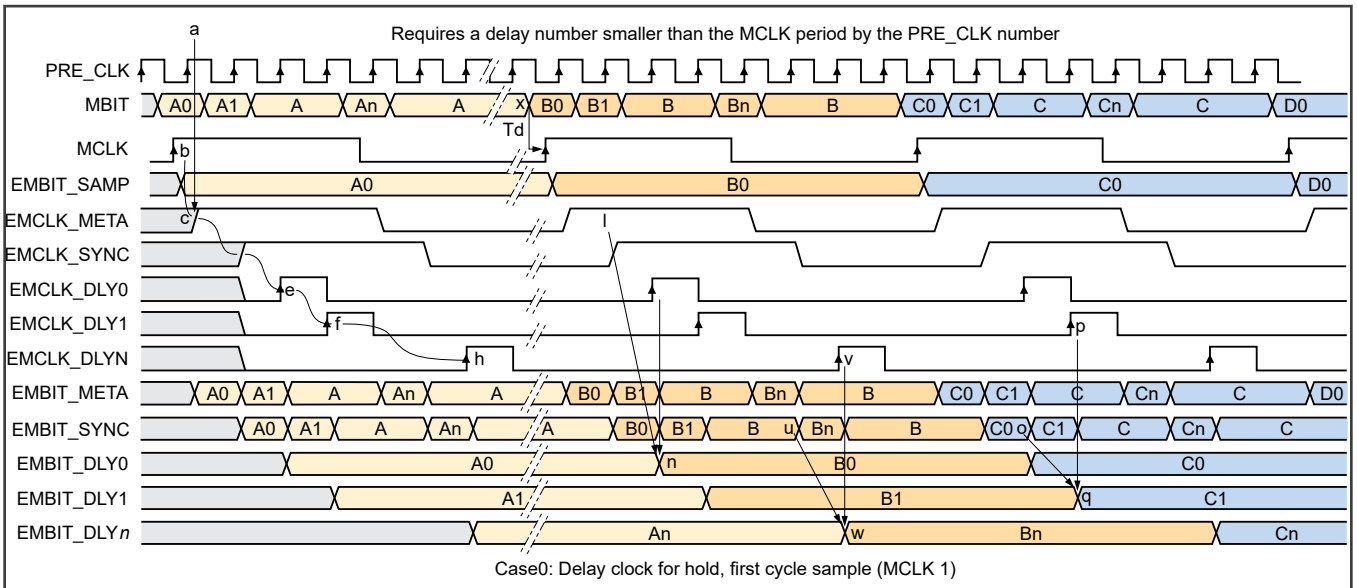


Figure 215. Modulator bitstream delay case 0

47.3.2.3.4.2 Modulator bitstream delay case 1

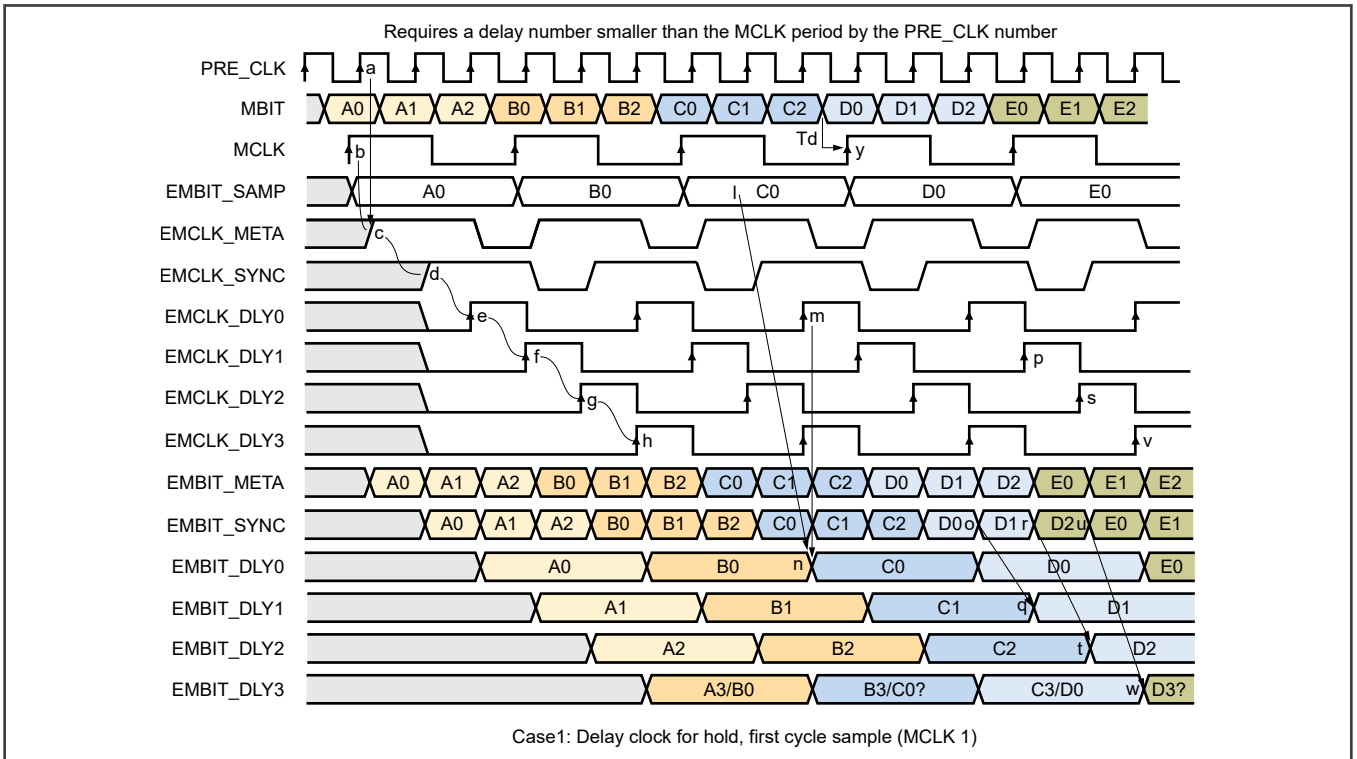


Figure 216. Modulator bitstream delay case 1

47.3.2.3.4.3 Modulator bitstream delay case 2

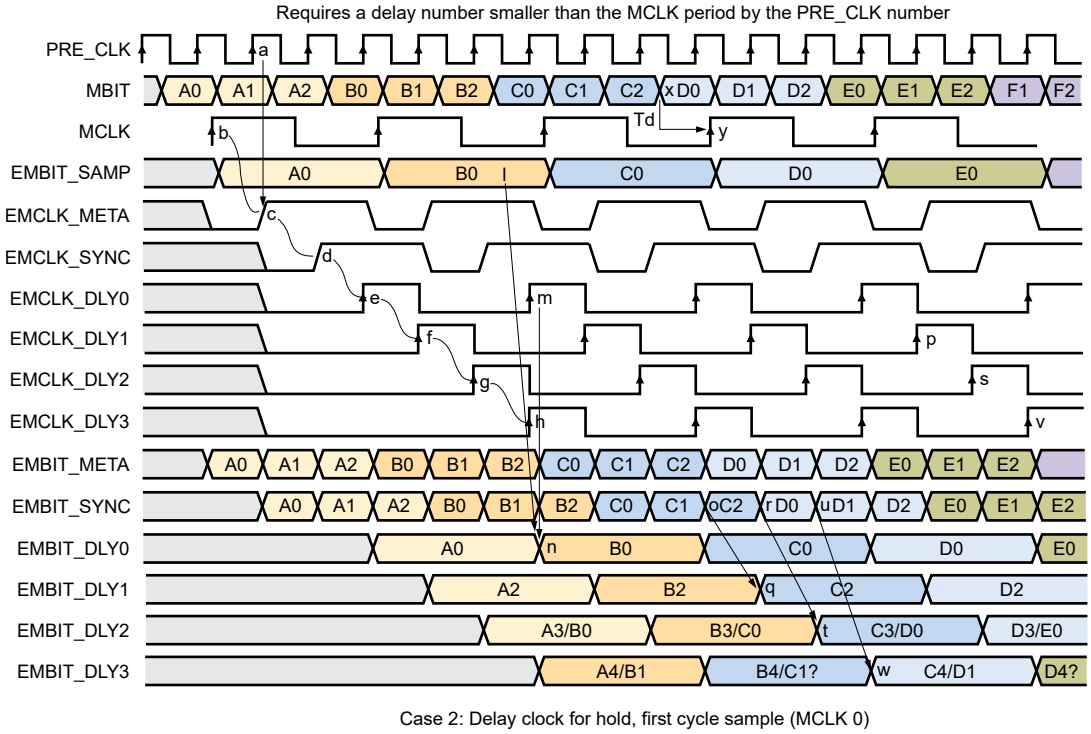


Figure 217. Modulator bitstream delay case 2

47.3.2.3.4.4 Modulator bitstream delay case 3

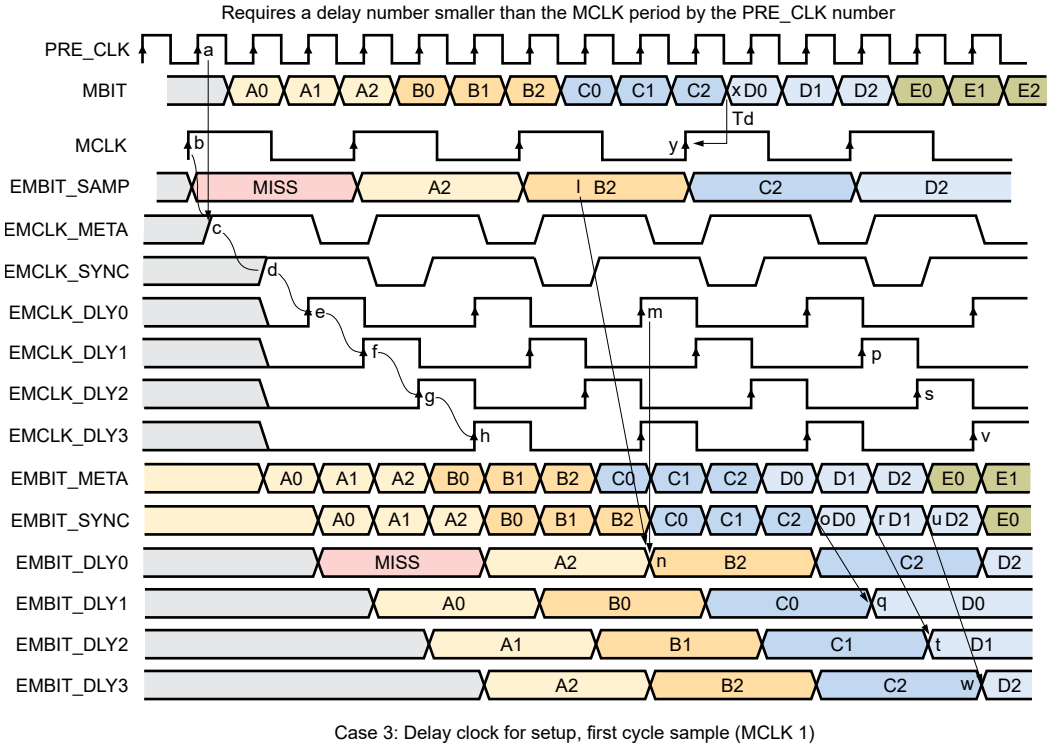


Figure 218. Modulator bitstream delay case 3

47.3.2.3.4.5 Modulator bitstream delay case 4

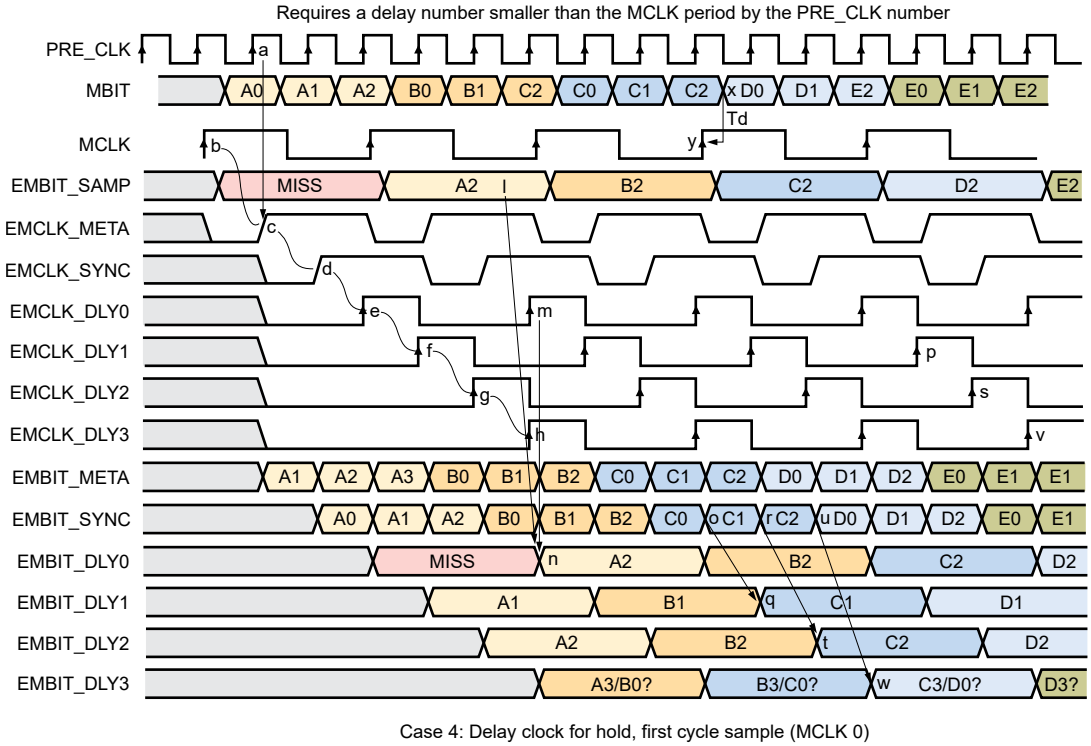


Figure 219. Modulator bitstream delay case 4

47.3.2.3.5 EM clock input bitstream sampling

See EM clock input bitstream sampling: settings for the values you need to write to SINC register fields for EM clock input bitstream sampling and see EM clock input bitstream sampling: diagram for the related bitstream sampling diagram.

47.3.2.3.5.1 EM clock input bitstream sampling: settings

Table 398. EM clock input bitstream sampling: settings

Action	Result
Writing 00b to C*n*CFR[IBFMT]	Selects the CIC filter input from the selected modulator bitstream input (EMBIT_SEL[*n*]), which the EMCLK_SEL[*n*] clock samples to meet the I/O timing requirement and deliver for further decode.
If you write 011b to C*n*CFR[ICSEL]	The selected external clock (EMCLK_SEL) becomes a dedicated modulator clock, MCLK[*n*], for this channel.
If you write 111b to C*n*CFR[ICSEL]	The selected external clock (EMCLK_SEL) becomes a shared group clock from an adjacent channel.
If you write 000b, 001b, or 010b to C*n*CFR[ICSEL]	The selected external clock (EMCLK_SEL) becomes one of the three internal clocks that are routed as modulator clock outputs.

Table continues on the next page...

Table 398. EM clock input bitstream sampling: settings (continued)

Action	Result
If you write 00b to C*n*CFR[IBSEL]	The selected external modulator bitstream (EMBIT_SEL) becomes a dedicated external modulator bitstream (MBIT[*n*]) for this channel.
If you write 01b to C*n*CFR[IBSEL]	The selected external modulator bitstream (EMBIT_SEL) becomes a dedicated internal modulator bitstream (inp[*n*]) for this channel.
If you write 11b to C*n*CFR[IBSEL]	The selected external modulator bitstream (EMBIT_SEL) becomes a shared group bitstream from an adjacent channel, *n*-1.
Writing an appropriate value to C*n*CFR[ICESEL]	Helps you select the type of edge to sample EMBIT_SEL.
Writing an appropriate value to C*n*ACFR[IBDLY]	Helps you select the modulator bitstream delay function.

NOTE

You must configure the prescaler correctly to sample EMCLK_SEL according to C*n*CFR[ICESEL]. If you configure C*n*CFR[ICESEL] as a single-edge sample, you must program the prescaler to be 3x faster than EMCLK_SEL; if you configure C*n*CFR[ICESEL] as a dual-edge sample, you must program the prescaler to be 6x faster than EMCLK_SEL.

See the following for more information:

- Modulator bitstream and channel clock selection
- Modulator bitstream programmed delay
- Bitstream source selection

47.3.2.3.5.2 EM clock input bitstream sampling: diagram

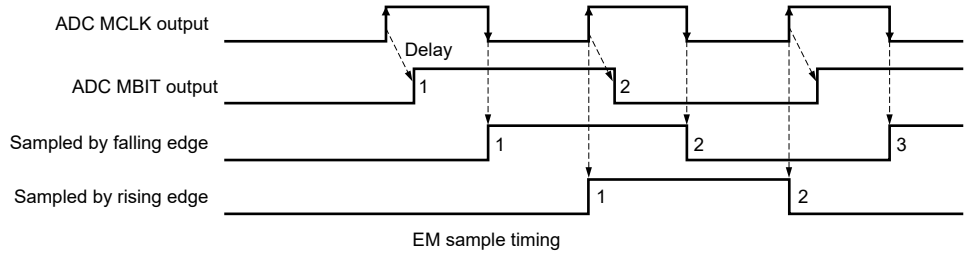


Figure 220. EM clock input bitstream sampling: diagram

47.3.2.3.6 MM clock input bitstream sampling

See MM clock input bitstream sampling: settings for the values you need to write to SINC register fields for MM clock input bitstream sampling and see MM clock input bitstream sampling: diagram for the related bitstream sampling diagram.

47.3.2.3.6.1 MM clock input bitstream sampling: settings

Table 399. MM clock input bitstream sampling: settings

Action	Result
Writing 01b to C*n*CFR[IBFMT]	Helps you select the input bitstream format as external Manchester code. The modulator clock is recovered from the bitstream.
Writing an appropriate value to C*n*CFR[ICESEL]	Helps you select the input clock edge as rising or falling, with only 001b and 010b are valid values (see MM clock input bitstream sampling: diagram).
Writing 00b to C*n*CFR[IBSEL]	Helps you define the selected external modulator bitstream (EMBIT_SEL) as a dedicated external modulator bitstream (MBIT[*n*]) for this channel.
Writing 01b to C*n*CFR[IBSEL]	Helps you define the selected external modulator bitstream (EMBIT_SEL) as a dedicated internal modulator bitstream (INP[*n*]) for this channel.
Writing 11b to C*n*CFR[IBSEL]	Helps you define the selected external modulator bitstream (EMBIT_SEL) as a shared group bitstream from an adjacent channel, *n*-1.
Writing an appropriate value to C*n*MPDATA	Helps you define the correct threshold for the Manchester decoder. The threshold value must lie between the long pulse width and the short pulse width, and the prescaler clock counts this MBIT[*n*] pulse width. As shown in MM clock input bitstream sampling: diagram, if the previous short pulse is not recovered correctly, the first long pulse is considered as the first valid bit used for whole packet synchronization.

For more on the aforementioned settings, see the following:

- Modulator bitstream and channel clock selection
- Bitstream source selection

47.3.2.3.6.2 MM clock input bitstream sampling: diagram

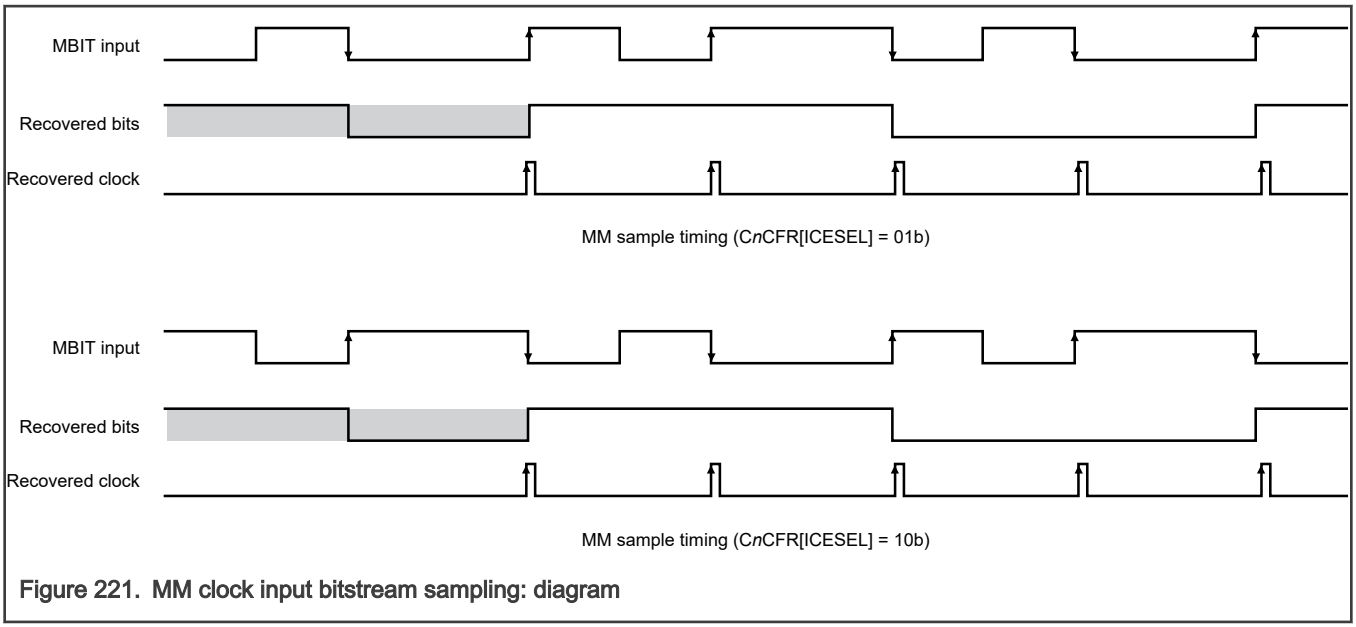


Figure 221. MM clock input bitstream sampling: diagram

47.3.2.3.7 PM clock input bitstream sampling

Table 400. PM clock input bitstream sampling

Action	Result
Writing 10b to CnCFR[IBFMT]	Selects the CIC filter input from Channel n Multipurpose Data (C0MPDATA - C4MPDATA).
Checking the value of CnSR[PSRDY]	Helps you determine if the next write is allowed. CnSR[PSRDY] = 1 indicates that the previously written data is processed and the field is waiting for the next PM write.
Writing to CnCFR[RDFMT]	Selects PM data as signed or unsigned.
Writing to Channel n Multipurpose Data (C0MPDATA - C4MPDATA)	Sends the lower 16 bits to the CIC filter. The PM clock generates automatically after a write event; IMCLK0 decides the clock rate. If a PM write is not allowed (CnSR[PSRDY] = 0 when CnCFR[IBFMT] = 10), writing to Channel n Multipurpose Data (C0MPDATA - C4MPDATA) returns a bus error.

NOTE

Modulator clocks are automatically generated when you write data to Channel n Multipurpose Data (C0MPDATA - C4MPDATA). Because of conversion latency, you must write an extra two dummy data samples to provide extra clocks after the conversion is complete.

47.3.2.3.8 SM clock input bitstream sampling

See SM clock input bitstream sampling: settings for the values you need to write to SINC register fields for SM clock input bitstream sampling and see SM clock input bitstream sampling: diagram for the related bitstream sampling diagram.

47.3.2.3.8.1 SM clock input bitstream sampling: settings

Table 401. SM clock input bitstream sampling: settings

Action	Result
Writing 11b to C*n*CFR[IBFMT]	Selects the CIC filter input from Channel n Multipurpose Data (C0MPDATA - C4MPDATA). The entire 32-bit serial data is shifted to the CIC filter. SM clocks are auto generated after a write event, and IMCLK0 decides the clock rate.
Checking the value of C*n*SR[PSRDY]	Determines if the next SM write is allowed. C*n*SR[PSRDY] = 1 indicates that the previously written data is processed and the field is waiting for the next SM write.
Writing to Channel n Multipurpose Data (C0MPDATA - C4MPDATA) for SM	Returns a bus error if an SM write is not allowed (C*n*SR[PSRDY] = 0), and no bitstream remains to be shifted after conversion finishes (C*n*CFR[IBFMT] = 11b).

NOTE

After conversion, the remaining bits are retained in Channel n Multipurpose Data (C0MPDATA - C4MPDATA). To reset them, you could toggle C*n*CR[CHEN] or MCR[RST].

47.3.2.3.8.2 SM clock input bitstream sampling: diagram

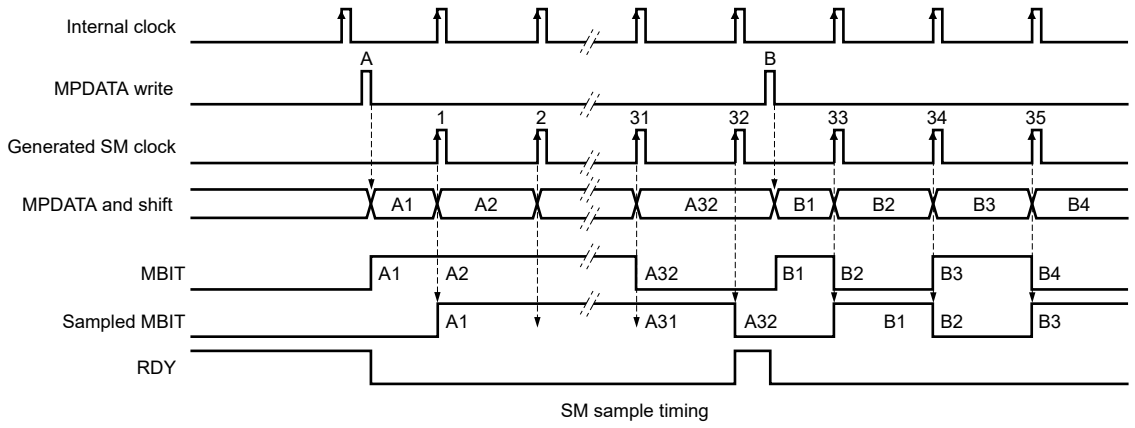


Figure 222. SM clock input bitstream sampling: diagram

47.3.2.4 Conversion process

47.3.2.4.1 Modes and behaviors

After a conversion process triggers:

- The decimation filters are activated.
- NIS[COC*n*] becomes 1 after the conversion completes.

You could use C*n*DR[PFCM] to select one of the conversion modes shown in the following table.

Table 402. Modes and behaviors

Mode	Behavior of the conversion process in this mode
Single (C*n*DR[PFCM] = 00b)	• Each conversion requires a conversion trigger. • A single trigger starts only one conversion.
Continuous (C*n*DR[PFCM] = 01b)	• After a trigger is detected, the conversion process continues until the next trigger event. • The conversion process also continues after the first sample. You can perform a series of conversions at the rate of one sample per decimation clock cycle, which equals the value of C*n*DR[PFOSR] + 1. For details on subsequent samples, see Total conversion time. • The next trigger cancels the current conversion and restarts it.
Always (C*n*DR[PFCM] = 10b)	• After a trigger is detected, the conversion process continues until it is disabled. • Conversions also continue after the first sample. You can perform a series of conversions at the rate of one sample per decimation clock cycle, which equals the value of C*n*DR[PFOSR] + 1. For details on subsequent samples, see Total conversion time. • The next trigger event is ignored.
Fixed-Number (C*n*DR[PFCM] = 11b)	• After a trigger is detected, the conversion process continues until the required conversion number is achieved or until the next trigger event. • The conversion process stops when NIS[CHF*n*] becomes 1, so you can configure C*n*CFR[FIFOWMK] to determine the number of needed conversions. • The conversion process also continues after the first sample. You can perform a series of conversions at the rate of one sample per decimation clock cycle, which equals the value of C*n*DR[PFOSR] + 1. For details on subsequent samples, see Total conversion time. • The next trigger cancels the current conversion and restarts it.

47.3.2.4.2 Completion

Following are some of the events that take place after a conversion completes:

- The new sample value is transferred to the FIFO, and you can get the FIFO content by reading Channel n Result Data (C0RDATA - C4RDATA).
- NIS[COC*n*] becomes 1 and the FIFO data is available.
- NIS[CHF*n*] becomes 1 if the FIFO from the PF channel *n* has overflowed its watermark level and the FIFO data is available.
- A conversion complete and CHF interrupt are asserted if NIE[COCIE*n*] = 1 or NIE[CHFIE*n*] = 1.
- A DMA request and CHF are asserted if C*n*CR[DMAEN] = 1.

Additionally, you can read the current conversion data directly from Channel n Result Data (C0RDATA - C4RDATA) if FIFO is disabled.

47.3.2.4.3 Discontinuation

Conversion involves the following steps:

- CIC calculation
- Shift
- DC removal
- Bias
- Push FIFO and interrupt

You could abort a conversion by using any of these techniques:

- Disable
- Next Trigger
- Stop

This is what you could do to abort conversions, by using different register fields, modes, and triggers:

- To abort a conversion in progress, disable the PF by writing 0 to C*n*CR[PFEN]. However, in case you cancel the CIC filter by using C*n*CR[PFEN], the SCD and CAD blocks can still keep working.
- To abort a conversion along with its subfunctions (PF, CAD, and SCD), disable the channel by writing 0 to C*n*CR[CHEN].
- To abort or disable all the conversions globally, write 0 to MCR[MEN]. The FIFO retains the values of the previous conversion result that had not been read.
- In Single and Continuous conversion modes, if you select an edge trigger, a new trigger aborts the current conversion and restarts a new one.
- If you select a level trigger, a deasserted software or hardware trigger cancels the current conversion.
- In Stop mode, the current conversion is canceled.

If an abort occurs after the CIC filter has performed its function, you could:

- Use the Disable or Stop technique to cancel the current conversion immediately.
- Use the Next Trigger technique for the current conversion to continue the shift and bias operations and save the result to the FIFO. At the same time, the CIC can process the next conversion.

NOTE

For a motor use case, you must calculate the maximum speed (trigger frequency) to ensure that the required conversion number and the next trigger do not overlap. The margin must be at least two modulator clocks.

47.3.2.5 Conversion timing

47.3.2.5.1 Total conversion time

Table 403. Total conversion time

In this mode:	The conversion time is:
Single	The value derived using this formula for a single conversion after the trigger event is over: ORD × OSR × MOD_CLK period
Continuous	The value derived using this formula for the first 24-bit sample after the trigger event is over: ORD × OSR × MOD_CLK period However, the subsequent 24-bit samples are available after the OSR × MOD_CLK period.
Always	
Fixed-Number	

where MOD_CLK period = 1 ÷ the frequency of MOD_CLK.

During a conversion, if you opt to resynchronize the conversion by providing a trigger, the first conversion result is available after the ORD × OSR × MOD_CLK period.

However, if you opt for a run-time phase compensation for a channel, SINC does not support it. You must provide a hardware trigger delay (for example, a PDB block) at the chip level. SINC does not support a software trigger delay.

47.3.2.5.2 Trigger alignment

When a trigger asserts, the recovered modulator clock samples the trigger event (see MOD_CLK in Bitstream source selection). At the same time, the modulator clock samples the MBIT[*n*] signal (see External signals). Therefore, the first valid bitstream to CIC is the clock cycle during which the trigger is asserted.

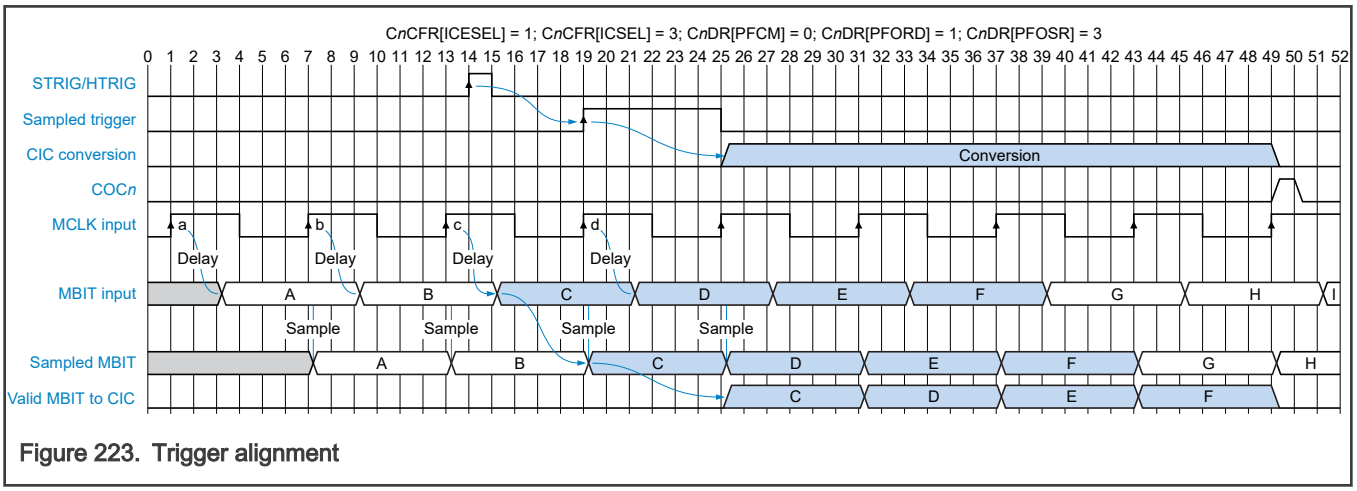


Figure 223. Trigger alignment

47.3.2.6 Trigger timing

47.3.2.6.1 Single mode software edge trigger

As illustrated in the following figure, when a software edge trigger event occurs in Single mode:

- Triggers a and b are ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger c starts conversion C1; trigger d starts conversion D1.
- Trigger e starts conversion e1 and trigger f cancels it before the conversion is complete. Simultaneously, trigger f starts (or resynchronizes) conversion F1.
- Trigger g starts conversion g1 but deassertion of the "Enable" signal cancels it.

See Total conversion time that defines the formula for Single mode conversion time.

NOTE

- The "Enable" signal is a combined AND logic of C*n*CR[PFEN] and SR[CHRDY*n*].
- You write 1 to MCR[STRIG*n*] and SINC writes 0 to this field for a software edge trigger case.

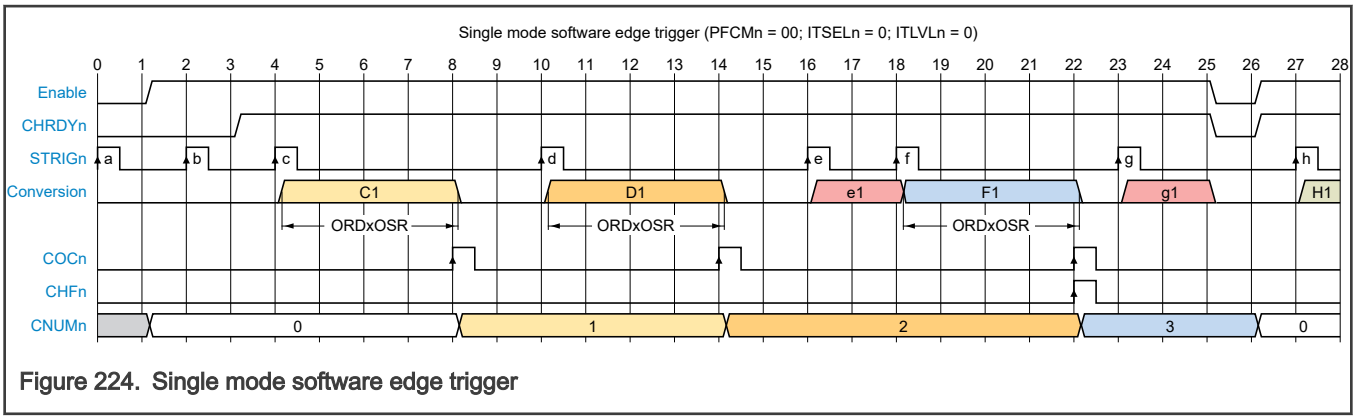


Figure 224. Single mode software edge trigger

47.3.2.6.2 Single mode hardware edge trigger

As illustrated in the following figure, when a hardware edge trigger event occurs in Single conversion mode:

- Triggers a and b are ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger c starts conversion C1; trigger d starts conversion D1.
- Trigger e starts conversion e1, but trigger f cancels it before completion. Simultaneously, trigger f starts (or resynchronizes) conversion F1.
- Trigger g starts conversion g1 but deassertion of the "Enable" signal cancels it.

The external hardware both asserts and deasserts the HTRIG[*n*] signal (see External signals); however, only the rising edge of the signal is accepted as the startup event for conversion. For example, trigger d exists several times, but the fall of trigger g does not cancel conversion D1.

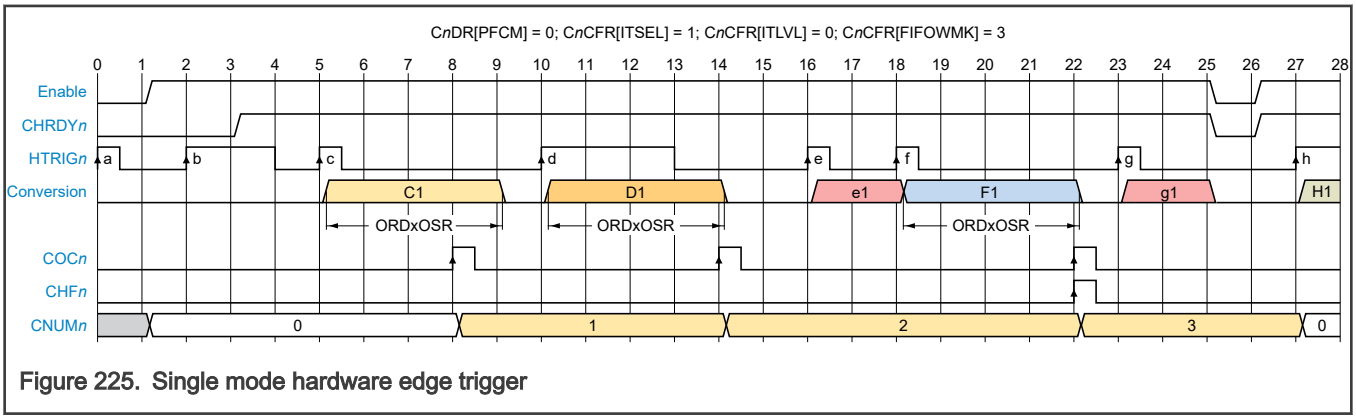


Figure 225. Single mode hardware edge trigger

47.3.2.6.3 Single mode hardware level trigger

As illustrated in the following figure, when a hardware level trigger event occurs in Single mode:

- Trigger a is ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger b starts conversion B1; trigger c starts conversion C1.
- Trigger d starts conversion d1 but the conversion cancels because trigger d deasserts before conversion completion.
- Trigger f starts conversion f1 but the conversion cancels because the "Enable" signal deasserts.

An external hardware asserts and deasserts HTRIG[*n*]. The rising of HTRIG[*n*] starts the conversion and the falling of HTRIG[*n*] cancels the conversion before its completion.

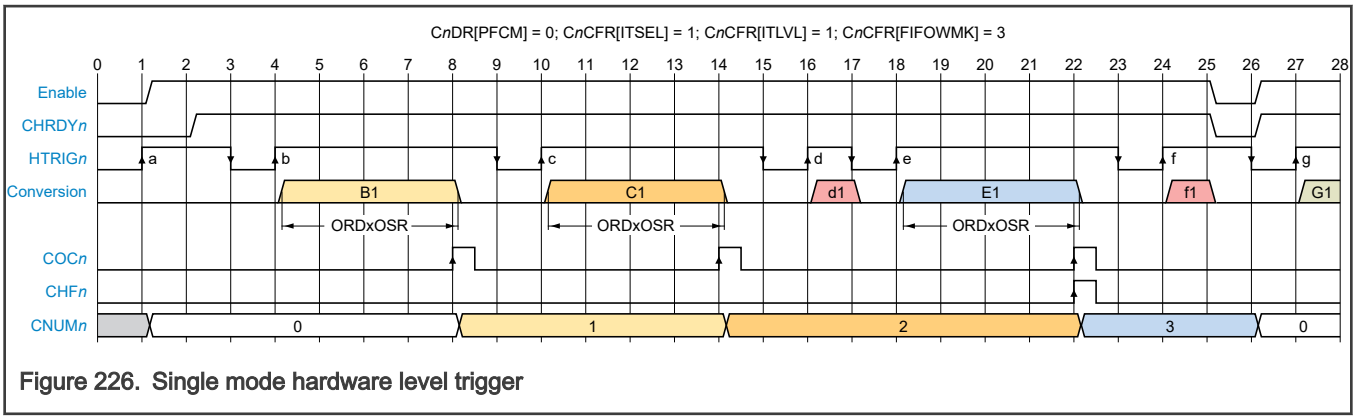


Figure 226. Single mode hardware level trigger

47.3.2.6.4 Continuous mode software edge trigger

In Continuous mode, conversions comprise a set of subconversions and continue until aborted by another event. For example, as shown in the following figure, conversion C comprises subconversions C1, C2, C3, c4, and so on; and conversion D comprises subconversions D1, D2, d3, and so on. When a software edge trigger event occurs in Continuous conversion mode:

- Triggers a and b are ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger c starts conversion C1 and trigger d starts conversion D1.
- The output delay time for C1 is defined as OSR×ORD. When conversion C1 comes to an end, conversion C2 starts automatically.
- The output delay time for C2 is defined as OSR, and so is the output delay time for conversion C3.
- Trigger d cancels conversion c4 and simultaneously resynchronizes conversion D1.
- Writing 0 to C*n*CR[PFEN] cancels conversion d3.

You write 1 to MCR[STRIG*n*] and SINC writes 0 to this field for a software edge trigger event in Continuous mode.

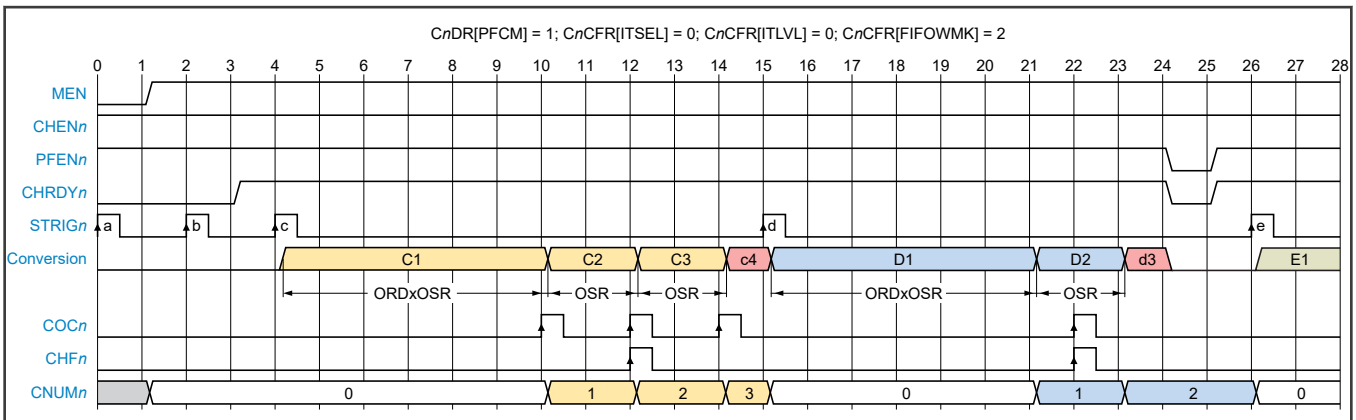


Figure 227. Continuous mode software edge trigger

47.3.2.6.5 Continuous mode hardware edge trigger

As illustrated in the following figure, when a hardware edge trigger event occurs in Continuous mode:

- Triggers a and b are ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger c starts conversion C1, and hardware trigger d starts conversion D1.
- The output delay time for C1 is defined by OSR×ORD. After conversion C1 comes to an end, conversion C2 starts automatically.

- The output delay time for C2 is defined as OSR, and so is the output delay time for C3.
- Trigger d cancels conversion c4 and simultaneously resynchronizes conversion D1.
- Writing 0 to C*n*CR[PFEN] cancels conversion d4.

The external hardware asserts and deasserts HTRIG[*n*]. However, only the rising of HTRIG[*n*] starts the conversion. For example, the hardware trigger d exists several times, but the fall of trigger d does not cancel conversion D1. Hardware trigger b asserts before SR[CHRDY*n*] becomes 1 and remains asserted after SR[CHRDY*n*] becomes 1, but no conversion starts.

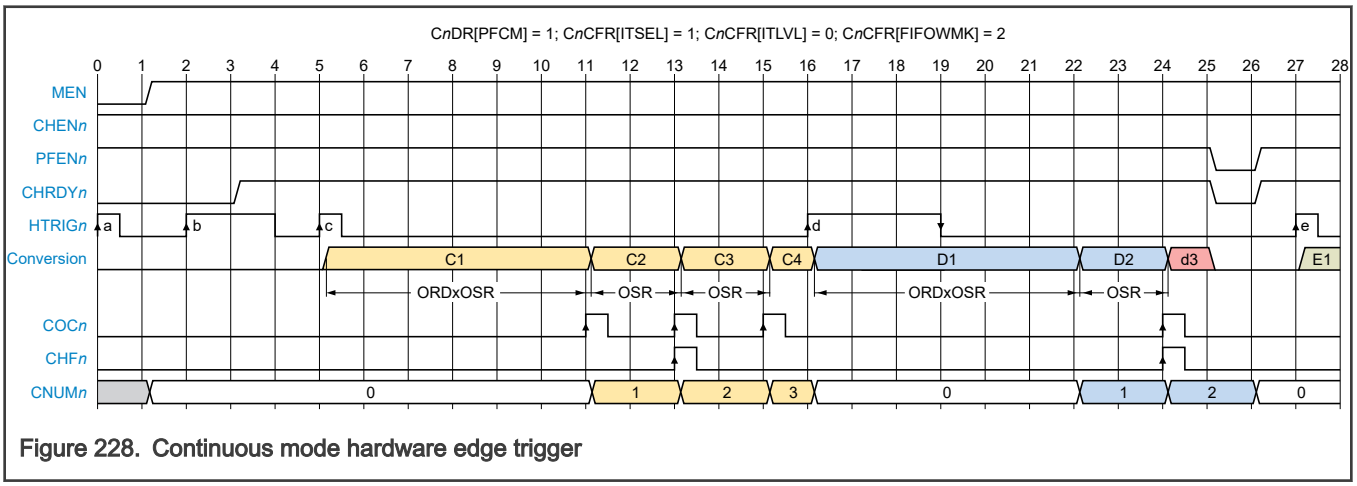


Figure 228. Continuous mode hardware edge trigger

47.3.2.6.6 Continuous mode software level trigger

As illustrated in the following figure, when a software level trigger event occurs in Continuous mode:

- Triggers a and b are ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger c starts conversion C1; trigger d starts conversion D1.
- The output delay time for C1 is defined as OSR×ORD. After conversion C1 comes to an end, conversion C2 starts automatically.
- The output delay time for C2 is defined as OSR, and so is the output delay time for C3.
- Trigger c deassertion cancels conversion c4.
- Signal "Enable" deassertion cancels conversion d3.

For a software level trigger event in Continuous mode, you can write both 1 (to start the conversion process) and 0 (to stop the conversion process) to MCR[STRIG*n*] if SR[CHRDY*n*] is 1. SINC can also write 0 to MCR[STRIG*n*] to cancel the conversion. The following example shows writing 0 to MCR[STRIG*n*] cancels conversion c4.

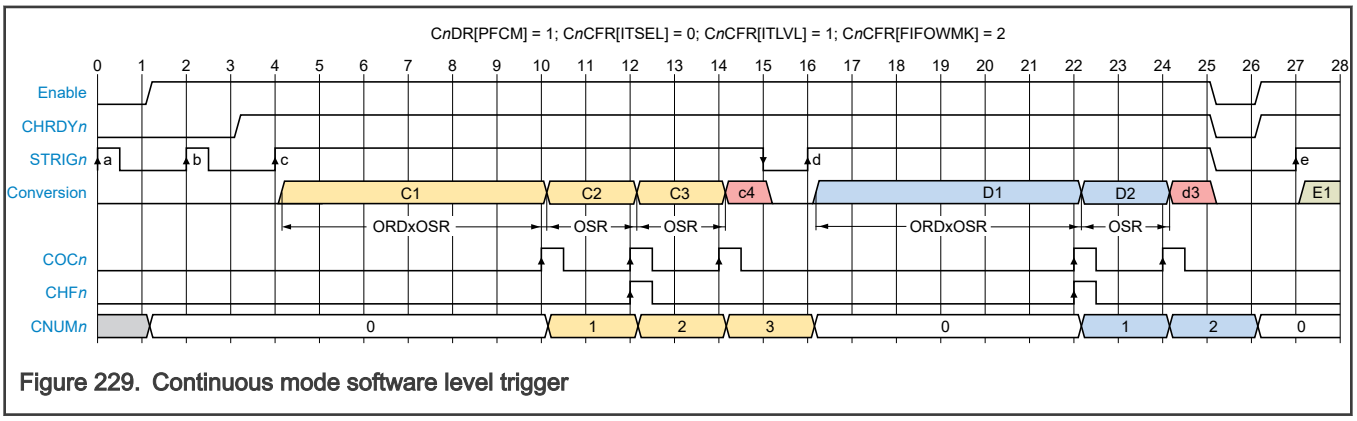


Figure 229. Continuous mode software level trigger

47.3.2.6.7 Continuous mode hardware level trigger

As illustrated in the following figure, when a hardware level trigger event occurs in Continuous mode:

- Triggers a and b are ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger c starts conversion C1; and trigger d starts conversion D1.
- The output delay time for C1 is defined as OSR×ORD. When conversion C1 comes to an end, conversion C2 starts automatically.
- The output delay time for C2 is defined as OSR, and so is the output delay time for conversion C3.
- Trigger c deassertion cancels conversion c4.
- Signal "Enable" deassertion cancels conversion d3.

An external hardware asserts and deasserts HTRIG[*n*]. It asserts HTRIG[*n*] to start the conversion process and deasserts it to stop the conversion process. The following example shows that HTRIG[*n*] deassertion cancels conversion c4.

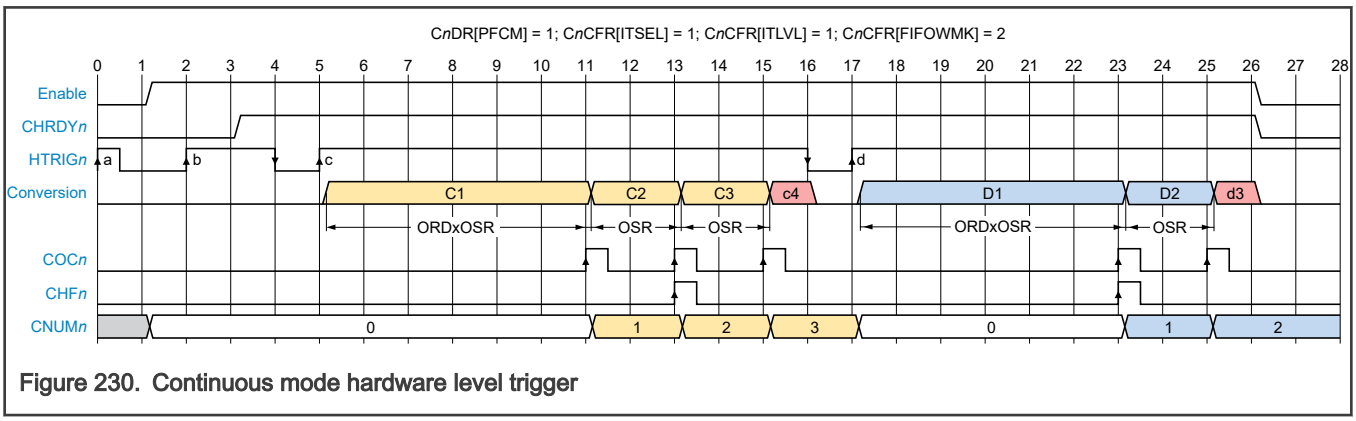


Figure 230. Continuous mode hardware level trigger

47.3.2.6.8 Always mode hardware edge trigger

As illustrated in the following figure, when a hardware edge trigger event occurs in Always mode:

- Triggers a and b are ignored because SR[CHRDY*n*] = 0 (not ready for conversion).
- Trigger c starts conversion C1; trigger e starts conversion D1.
- The output delay time for C1 is defined as OSR×ORD. After conversion C1 comes to an end, conversion C2 starts automatically.
- The output delay time for C2 is defined as OSR, and so is the output delay time for conversion C3.
- Trigger d is ignored.

After the conversion process starts in this mode, the only way to stop it is by writing 0 to MCR[MEN], C*n*CR[CHEN], or C*n*CR[PFEN].

An external hardware both asserts and deasserts HTRIG[*n*].

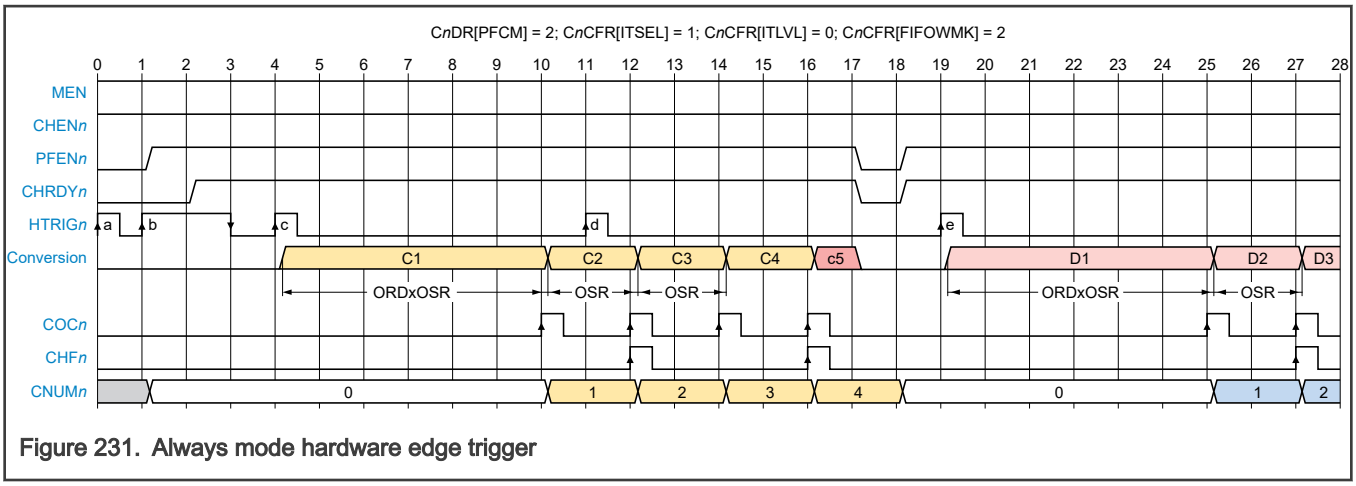


Figure 231. Always mode hardware edge trigger

47.3.2.7 PF

SINC's PF comprises the following:

- A CIC filter
- An optional HPF for DC removal operations
- A shift
- An optional bias

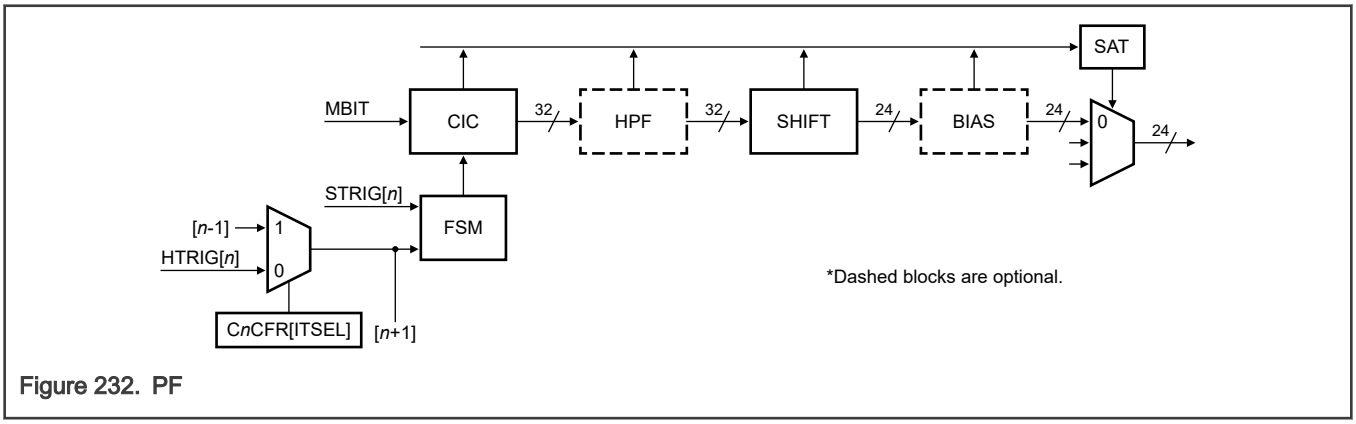


Figure 232. PF

47.3.2.7.1 CIC filter

CIC filters:

- Consist of one or more integrator and comb filter pairs.
- Represent an optimized implementation of low-pass filters (LPFs) with programmable, multiple stages.
- Are widely used because they can be implemented using just adders and subtractors.
- Output raw 32-bit data and deliver it to the HPF (or to the PF shift if HPF is not present).
- Select a CIC filter input, decode modulator bitstream, and decode modulator clock from the recovery block.

The following figure shows the architecture of a three-stage CIC filter, comprising integrator sections followed by comb sections downsampled.

See CnDR[PFORD] for information on PF order and CnDR[PFOSR] for information on PF OSR.

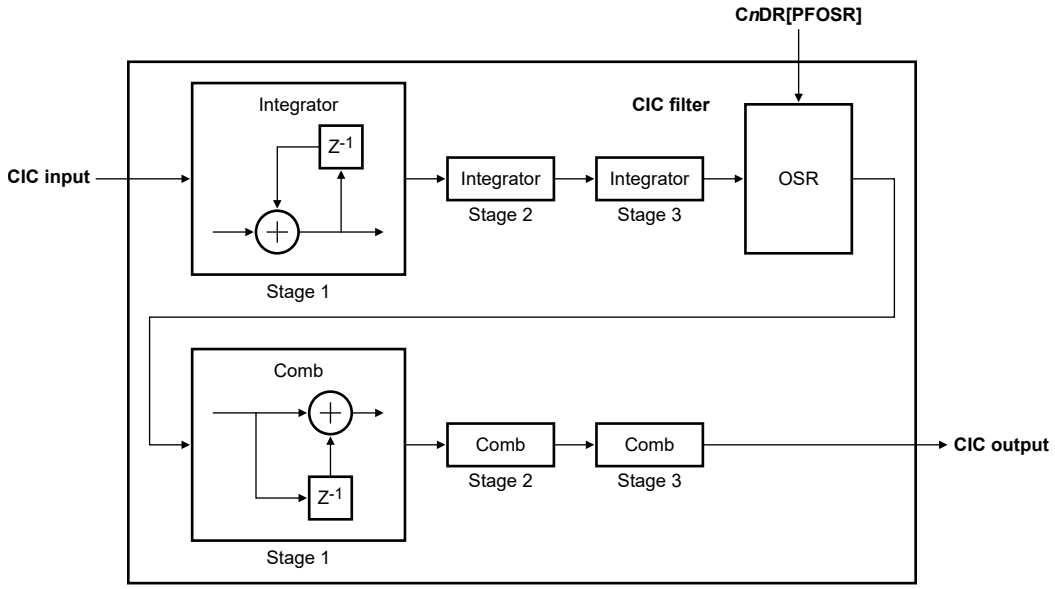


Figure 233. CIC filter

47.3.2.7.1.1 CIC filter transfer function

You can calculate the transfer function of the CIC filter, when C*n*DR[PFORD] is any value except 0, based on the following equation.

$$H(Z) = \frac{Y(Z)}{X(Z)} = \left[\frac{1-Z^{-OSR}}{1-Z^{-1}}\right]^{ORD}$$

Equation 21. CIC filter transfer function

You can see an example of Frequency Response in Frequency response

You can get CIC output effective data width by below equation. Where OSR are described at Equation for OSR.

when C*n*CFR[RDFMT]=0b(signed):

WDITH = log2(OSR^^C*n*DR[PFORD]+1)+1

when C*n*CFR[RDFMT]=1b(unsigned):

WDITH = log2(OSR^^C*n*DR[PFORD]+1)

Equation 22. Equation for CIC effective data width

47.3.2.7.1.2 FastSinc filter transfer function

You can calculate the transfer function of the FastSinc filter, when C*n*DR[PFORD] = 0, based on the following equation.

$$H(Z) = \frac{Y(Z)}{X(Z)} = \left[\frac{1-Z^{-OSR}}{1-Z^{-1}}\right]^{2} * \left[1+Z^{-2*OSR}\right]$$

Equation 23. FastSinc filter transfer function

You can get CIC output effective data width by below equation. Where OSR are described at Equation for OSR

when C*n*CFR[RDFMT]=0b(signed):

WDITH = log2(OSR^^2*2+1)+1

when C*n*CFR[RDFMT]=1b(unsigned):

WDITH = log2(OSR^^2*2+1)

Equation 24. Equation for CIC effective data width

47.3.2.7.1.3 CIC effective data width

You can see an example of Frequency Response in Frequency response

You can get example CIC output effective data width by Table 404. Where OSR are described at Equation for OSR

Table 404. data width vs OSR and ORD

OSR	C*n*CFR[RDFMT]=0b(signed)				C*n*CFR[RDFMT]=1b(unsigned)			
	PFORD=1	PFORD=2	PFORD=0 (fastsinc)	PFORD=3	PFORD=1	PFORD=2	PFORD=0 (fastsinc)	PFORD=3
16	6	10	11	14	5	9	10	13
32	7	12	13	17	6	11	12	16
64	8	14	15	20	7	13	14	19
128	9	16	17	23	8	15	16	22
256	10	18	19	26	9	17	18	25
1291	12	22	23	33(not support)	11	21	22	32
1626	12	23	24	34(not support)	11	22	23	33(not support)

47.3.2.7.2 Trigger events

The following table shows trigger cases based on the configuration of PF conversion modes (C*n*DR[PFCM]), input trigger selection (C*n*CFR[ITSEL]), and input trigger level type (C*n*CFR[ITLVL]).

NOTE

C*n*CFR[ITSEL] = 11b indicates that a grouped trigger shared with an adjacent channel behaves in the same way as a hardware trigger (01b).

Table 405. Trigger events

C*n*DR[PFCM]	C*n*CFR[ITSEL]	C*n*CFR[ITLVL]	Case	Functioning
00	00	0	Single mode software edge trigger	• Writing 1 to MCR[STRIG*n*] starts the conversion process. In case of an ongoing conversion, writing 1 to this field cancels it and starts a new conversion. • MCR[STRIG*n*] automatically becomes 0 after the conversion starts.

Table continues on the next page...

Table 405. Trigger events (continued)

CnDR[PFCM]	CnCFR[ITSEL]	CnCFR[ITLVL]	Case	Functioning
	00	1	Single mode software level trigger	Reserved
	01	0	Single mode hardware edge trigger	• An external hardware trigger rising edge starts the conversion process. • The next trigger during conversion cancels the current conversion and restarts a new one.
	01	1	Single mode hardware level trigger	• An external hardware trigger rising edge starts the conversion process. • The hardware trigger deasserts during conversion and cancels the current conversion.
01	00	0	Continuous mode software edge trigger	• Writing 1 to MCR[STRIGn] starts the conversion process, which continues unless the filter is disabled. In case of an ongoing conversion, writing 1 to this field cancels it and starts a new conversion. • MCR[STRIGn] automatically becomes 0 after the conversion starts.
	00	1	Continuous mode software level trigger	• Writing from 0 to 1 to MCR[STRIGn] starts the conversion process, which continues unless the channel filter is disabled. • Writing 0 to MCR[STRIGn] cancels the current conversion.
	01	0	Continuous mode hardware edge trigger	• An external hardware trigger rising edge starts the conversion process, which continues unless the channel filter is disabled. • The next trigger cancels the current conversion and restarts a new one.
	01	1	Continuous mode hardware level trigger	• An external hardware trigger rising edge starts the conversion process, which continues unless the channel filter is disabled. • The hardware trigger deasserts and cancels the current conversion.

Table continues on the next page...

Table 405. Trigger events (continued)

CnDR[PFCM]	CnCFR[ITSEL]	CnCFR[ITLVL]	Case	Functioning
10	00	0	Always mode software edge trigger	Reserved
	00	1	Always mode software level trigger	Reserved
	01	0	Always mode hardware edge trigger	• An external hardware trigger rising edge starts the conversion process, which continues unless the channel filter is disabled. • The next trigger is ignored.
	01	1	Always mode hardware level trigger	Reserved
11	Any value	1	Always mode hardware level trigger	• Trigger behavior is the same as in Continuous mode for CnCFR[ITSEL] and CnCFR[ITLVL]. • The conversion process automatically stops when the FIFO watermark is asserted.

47.3.2.7.3 HPF transfer function for DC removal

An HPF performs the DC removal function. The CIC filter provides an input to the HPF and outputs 32-bit data, which is sent to the PF shift.

CnACFR[HPFA] specifies the HPF alpha coefficient or disables the HPF when it is 0.

The HPF restarts if CnCR[CHEN] changes from 0 to 1 (in case of a rise event) and the following transfer function specifies the HPF frequency response.

$$H(Z) = \frac{Y(Z)}{X(Z)} = \frac{1\text{-}Z^{-1}}{1\text{-}aZ^{-1}}$$

Equation 25. HPF transfer function for DC removal

NOTE

For DC removal, you can perform HPF gain compensation and use only the signed data format.

47.3.2.7.4 Shift

The CIC filter outputs raw 32-bit data. The higher part of that data is most likely to consist of zeros or sign bits. You must shift this data to attain 24-bit alignment.

If the raw CIC data contains more than 24 bits, then you must shift it right by CnCFR[PFSFT] to fill bits 23–0, and copy the result to data bits 31–8.

If the raw CIC data contains fewer than 24 bits, then you must shift it left by CnCFR[PFSFT] to fill bits 23–0, and copy the result to data bits 31–8.

For right shift, SINC rounds the result up (as if adding 0.5×LSB) before truncation.

Rounding is not necessary for a left shift. If the shift and rounding result is more than 24 bits, the result is saturated and SINC changes the corresponding saturation status field to 1.

47.3.2.7.4.1 Shift right block diagram

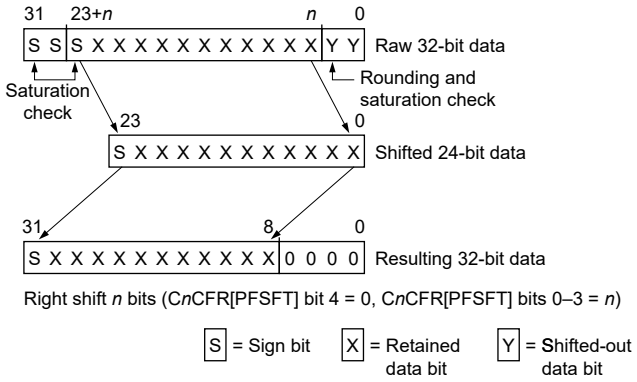


Figure 234. Shift right block diagram

47.3.2.7.4.2 Shift left block diagram

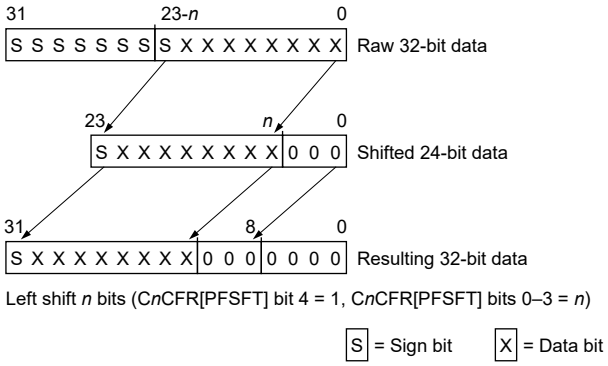


Figure 235. Shift left block diagram

47.3.2.7.5 Bias setting

Channel n Bias (C0BIAS - C4BIAS) specifies a bias offset for PF output, derives input from PF shift, and outputs the final result or value to FIFO and other compare blocks. The valid value is represented in the two's-complement format and the PF subtracts this bias value from the data.

47.3.2.8 SCD operations

An SCD is applied to each channel. Its purpose is to quickly respond if an analog signal is close to its maximum threshold value. The SCD can detect both short-circuit (overcurrent) and open-circuit (overvoltage) errors.

NOTE

An SCD does not support parallel modulator inputs.

Table 406. SCD operations

Using	You can
C*n*CR[SCDEN]	Write 0 to disable the SCD.
EIS[SCD*n*]	Check whether SINC detected a short circuit on filter channel *n*.
Channel n Debug (C0DBGR - C4DBGR) with an appropriate value of C*n*CR[DBGSEL]	Check SCD status anytime. When an SCD event occurs, the SCD continues to count for the same field until a bit toggle occurs and holds the counter for you to read.
EIE[SCDIEn]	Enable and disable the Short Circuit Detected interrupt for filter channel *n*.
C*n*PROT[SCDBK]	Enable and disable the automatic assertion of the BREAK_SCD signal (see External signals).
C*n*PROT[SCDLMT]	Specify the threshold value of the SCD counter. There exists an upcounting counter on each input channel that counts consecutive 0s or 1s on the modulator bitstream. The counter restarts in case of a bit toggle. If this counter reaches a threshold value (C*n*PROT[SCDLMT]), a short-circuit event is invoked.
C*n*PROT[SCDOP]	Program conversion options and control detection of the bit value (0, 1, or both) that increments the SCD counter.
C*n*PROT[SCDCM]	Program conversion modes and control when the SCD operates—when C*n*CR[CHEN] = 1 or when the PF is performing a conversion.

47.3.2.9 CAD operations

A CAD is applied to each channel. You must check the modulator clock input for absence to ensure that the correct conversion process and error detection are in progress.

NOTE

When a clock absence event occurs, the data conversion process and detectors provide incorrect data. You must manage this event and discard the given data when a clock absence is reported.

Table 407. CAD operations

Using field	You can
C*n*CR[CADEN]	Write 0 to disable CAD.
FIFOIS[CAD*n*]	Check whether SINC detected the absence of a clock on filter channel *n*. When a CAD event occurs, CAD continues to count until an overflow occurs and then latches the value until you clear the flag.
FIFOIE[CADIE*n*]	Enable and disable the Clock Absence interrupt for filter channel *n*.

Table continues on the next page...

Table 407. CAD operations (continued)

Using field	You can
C*n*PROT[CADBK]	Enable and disable the automatic assertion of the BREAK_CAD signal (see External signals).
C*n*PROT[CADLMT]	Specify the threshold value of the CAD counter. The internal counters use IMCLK0 (see Function clock); therefore, you must program MCR[PRESCALE] and MCR[MCLKDIV] correctly to ensure that MCLK0_OUT is faster than the expected modulator clock input. However, MCLK0_OUT must not be more than 15 times faster.

47.3.2.10 ZCD operations

A ZCD asserts when a filter channel changes sign. In case of the absence of a threshold register and ripples, the ZCD asserts multiple times.

Table 408. ZCD operations

Using field	You can
C*n*CFR[RDFMT]	Configure the field as signed for the ZCD to work correctly.
C*n*CR[ZCDEN]	Write 0 to disable ZCD.
NIS[ZCD*n*]	Check the status of ZCD.
NIE[ZCDIE*n*]	Enable and disable the Zero Cross Detected interrupt for filter channel *n*.
C*n*CFR[ZCOP]	Select the type of zero crossing—rise, fall, or both.

47.3.2.11 Limit detection operations

Each filter sample is compared to values in the limit registers. This comparison is based on the final conversion value with offset correction applied. C*n*HILMT[HILMT] and C*n*LOLMT[LOLMT] registers contain the corresponding values that are used to compare with the result. C*n*HILMT[HILMT] is used for result comparison with the high limit, and C*n*LOLMT[LOLMT] is used for result comparison with the low limit.

Table 409. Limit detection operations

Using field	You can
C*n*CR[LMTEN]	Write 0 to disable limit checking.
EIS[HLMT*n*], EIS[LLMT*n*], and EIS[WLMT*n*]	Check the status of high limit flag, low limit flag, and window limit flag, respectively.
EIE[HLMTIE*n*], EIE[LLMTIE*n*], and EIE[WLMTIE*n*]	Enable or disable the High Limit interrupt, Low Limit interrupt, and Window Limit Interrupt, respectively.
C*n*ACFR[ADMASEL]	Enable an alternative DMA and select the limit flag that triggers the DMA.
C*n*PROT[HLMTBK], C*n*PROT[LLMTBK], and C*n*PROT[WLMTBK]	Program to assert break signals BREAK_HIGH, BREAK_LOW, and BREAK_WIN, respectively (see External signals).
C*n*PROT[LMTOP]	Program a limit detection option.

47.3.3 FIFO

The FIFO is a first-in, first-out buffer that saves the final filtering results. Each FIFO has a depth of 8 entries, and each entry stores a 24-bit result. Each channel has one FIFO. SINC writes (pushes) data into the FIFO, and you read (pop) data from the FIFO by using Channel n Result Data (C0RDATA - C4RDATA).

The FIFO has a configurable watermark (C*n*CFR[FIFOWMK]). When the amount of data in the FIFO is greater than C*n*CFR[FIFOWMK], SINC requests a DMA, IRQ, or both, depending on NIE[CHFIE*n*] and C*n*CR[DMAEN].

When the FIFO is full, SINC discards a new filtered result and sets the overflow flag (FIFOIS[FOVF*n*]). Similarly, if you attempt a pop operation and the FIFO is empty, SINC does not return any valid data and sets the underflow flag (FIFOIS[FUNF*n*]). When an overflow or underflow occurs, SINC sets the aforementioned flags and can generate an error interrupt you write 1 to the corresponding field in FIFO And CAD Error Interrupt Enable (FIFOIE).

When FIFO is enabled (C*n*CR[FIFOEN] = 1), MCR.MEN = 1, but C*n*CR[CHEN] = 0, you can still read the FIFO to flush it.

When you reset the FIFO by writing 1 to MCR[RST], SINC fills all FIFO entries with zeros and resets the push and pop pointers to their initial position.

Only a PF can serve as the input to a FIFO.

You can use either of the following methods to determine how much data remains in a FIFO:

- Read C*n*SR[FIFOAVIL].
- Write 9d to C*n*CR[DBGSEL], then read Channel n Debug (C0DBGR - C4DBGR).

47.3.4 Wake-up

SINC does not support an asynchronous interrupt to wake up the chip.

47.3.5 Details of debug data

Table 410. Details of debug data

When C*n*CR[DBGSEL] is:	C*n*DBGR[DBGDATA] contains:
0b	Final data from the PF (24 bits)
1b	Offset data (24 bits)
10b	Shifted data from the PF (24 bits)
11b	DC remover (HPF) data (32 bits)
100b	Raw data from the PF's CIC filter
110b	Historical data from short-circuit detection: • Bit 0 indicates the current count bit value. • Bits 1–8 indicate the current count. • Bits 9–16 indicate the previous count. • Bits 17–24 indicate the second-previous count. • Bits 25–31 are reserved.
111b	Data from the Manchester decoder: • Bits 0–7 indicate the number of decode errors (resync) that occurred after you enabled the channel. When this number exceeds 256, SINC resets it to 128.

Table continues on the next page...

Table 410. Details of debug data (continued)

When CnCR[DBGSEL] is:	CnDBGR[DBGDATA] contains:
	• Bits 8–18 indicate how long (in clock cycles) the current bit value has existed. • Bit 19 is the current bit value. • Bits 20–31 are reserved.
1000b	Data from CAD: • Bits 0–3 indicate the number of cycles during which SINC did not detect a modulator clock. SINC resets this number when SINC detects the modulator clock again. If it exceed the CnPROT[CADLMT], it will hold the value until software clear it. The maximum reportable number of clockless cycles is 15d (1111b). If more clockless cycles occur, the value in bits 0–3 remains 15d (1111b). • Bits 4–31 are zero.
1001b	Number of available entries in the FIFO: • Bits 0–7 indicate the number of available entries. • Bits 8–31 are zero.
1010b	Status of the parallel or serial data converter: • Bits 0–5 indicate the number of bits that SINC must still shift out as mod bits, for serial data converter. • Bits 6–7 are reserved. • Bits 8–31 are zero.

47.3.6 Modes of operation

Table 411. Modes of operation

Mode	Description
Normal	Default operational mode in which channels perform filtering operations. An external bitstream from the MBIT[*n*] signal (see External signals) supplies the input data and the MCLK_OUT[*n*] signal (see External signals) delivers the modulator clock to the external ADC.
Low-Power	Mode in which the chip gates the CPU clock, and if MCR[DOZEN] = 1, SINC functions are canceled. If the CPU is in Wait mode, SINC filters and protects the modulator bitstream data and can generate a wake-up event through a synchronous DMA or interrupt, depending on the values of MCR[MEN], CnCR[CHEN], CnCR[PFEN], and other protection block enable fields (CnCR[CADEN], CnCR[SCDEN], or CnCR[ZCDEN]). If the CPU clock enters very low power stop (VLPS) mode, the CPU clock is gated and a peripheral-bus stop acknowledge signal is raised if: • MCR[MEN] = CnCR[CHEN] = 1.

Table continues on the next page...

Table 411. Modes of operation (continued)

Mode	Description
	• Either C*n*CR[PFEN] or the other related protection enable fields (C*n*CR[CADEN], C*n*CR[SCDEN], or C*n*CR[ZCDEN]) are 1. You cannot generate DMAs and interrupts asynchronously to wake up the CPU clock.
Single	Mode in which you can control a typical motor using coils. For more on this mode, see Use case: Motor control with Single mode.
Continuous	Mode in which you can control audio applications and power meters. For more on this mode, see Use case: Audio system with Continuous mode, and Use case: Power meter with Continuous mode.
Fixed-Number	Mode in which you can control a typical motor and enhance this control by using techniques as such: • Adding a compensation filter to achieve better precision. • Flattening the passband. • Improving roll-off. For more on this mode, see Use case: Motor control with Fixed-Number mode.

47.3.7 Clocks

47.3.7.1 Overview

SINC has these types of clocks:

- Bus (see Bus clock)
- Function (see Function clock)
- Modulator (see Modulator clock)

The following sections provide:

- A block diagram of SINC clocks.
- An overview of the functioning and special clocking requirements of SINC clock types.

47.3.7.1.1 Clocks: block diagram

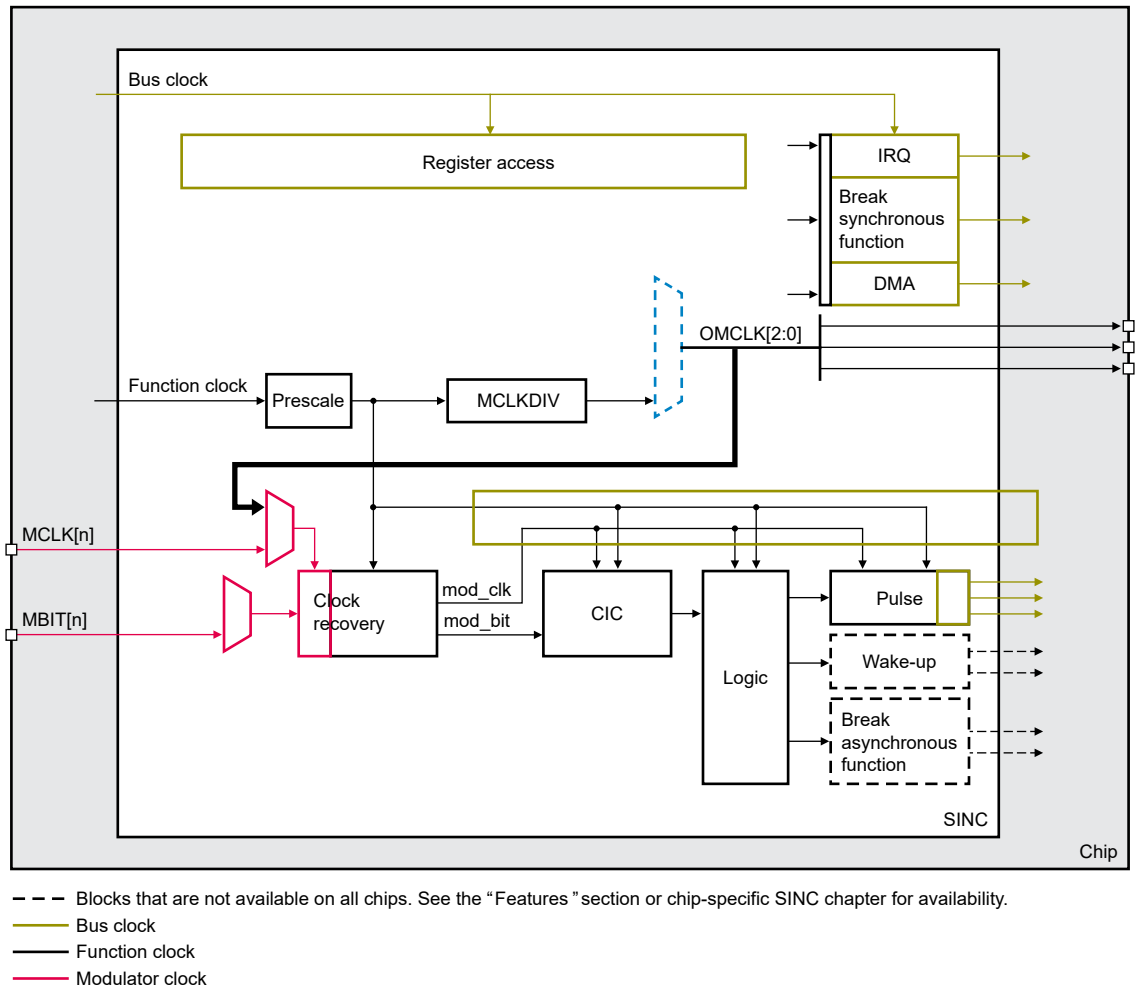


Figure 236. Clocks: block diagram

47.3.7.2 Bus clock

- Provides register access
- Allows control over low-power request signals, DMA, and interrupts

47.3.7.3 Function clock

The chip-provided function clock drives major functions, including:

- The prescaler
- The modulator clock divider (MCLKDIV)

The function clock is prescaled and divided to the correct operating range for SINC.

47.3.7.3.1 Main clock logic

The main clock logic, which is an integral part of the function clock:

- Is shared among all channels.
- Comprises a prescaler, MCLKDIV (divided by 2), and a clock output enable to external clocks.

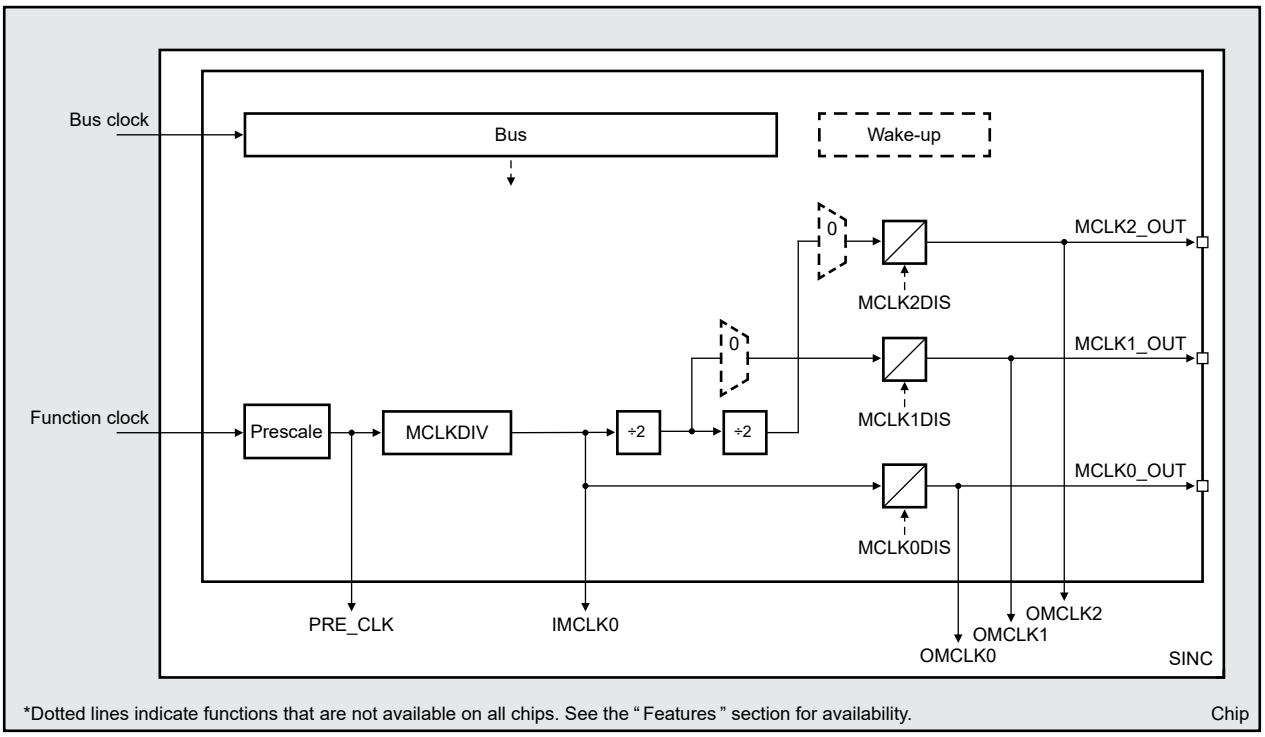


Figure 237. Main clock logic

47.3.7.3.2 Prescaler and output clocks

The prescaler supports power saving. It divides a high-frequency function clock (a major clock for other blocks) with an associated low-frequency clock. The prescaler divides the function clock by using MCR[PRESCALE], and then sends the clock to the divider and other blocks. See Main clock logic.

MCLKDIV sends the following three clocks to the chip for driving an external modulator ADC:

- MCLK_OUT0: Clock arrived at by adding 1 to the value of MCR[MCLKDIV].
- MCLK_OUT1: Clock arrived at by further dividing MCLK_OUT0 by 2.
- MCLK_OUT2: Clock arrived at by further dividing MCLK_OUT1 by 2.

These three aforementioned divided output clocks can be enabled or disabled by using MCR[MCLK0DIS], MCR[MCLK1DIS], and MCR[MCLK2DIS], respectively.

You must write an appropriate value to MCR[MCLKDIV]:

- For the target modulator clock to function properly. In some use cases, the modulator clock frequency must be certain times of the PWM frequency to use SINC's CIC filter.
- To obtain a 50% duty cycle, depending on the requirements of the external modulator ADC.

47.3.7.4 Modulator clock

A modulator clock:

- Is dedicated to each channel.
- Is provided by an external or system source.
- Can generate a digital output bitstream at a high data rate.

The SINC decimation filters reduce the sampling rate of the high-rate modulator output. Because of the latency of the decimation filters, each time a decimation filter is enabled or resynchronized, it takes one to four output samples (depending on the filter order) to reach a steady state producing valid output values.

47.3.7.4.1 Channel clock logic

The channel clock logic, which is a part of the modulator clock:

- Is dedicated to each channel individually.
- Comprises the following:
  - A normal decoder
  - A Manchester decoder
  - A clock source selection for these decoders
  - An MPDATA clock generator
- Includes the following:
  - An input asynchronous MCLK[*n*] selection, sample, or resynchronization
  - An MBIT[*n*] selection or sample
  - Manchester clock recovery
  - A clock select

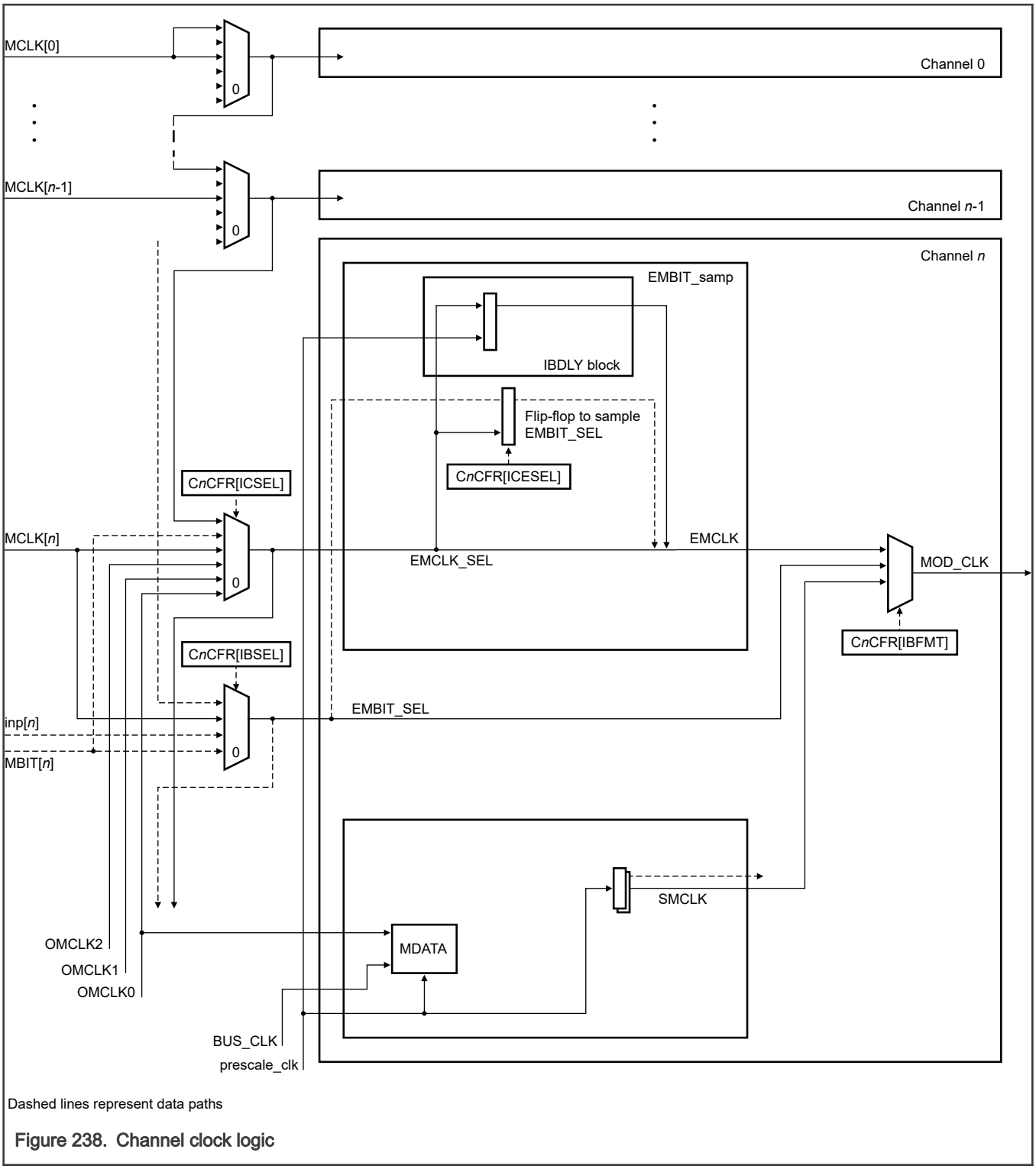


Figure 238. Channel clock logic

47.3.7.4.2 Prescaler considerations

You must program the prescaler to ensure that the predivided clock is:

- At least 3x faster than the MCLK[*n*] input for a single-edge sample.
- 6x faster than the MCLK[*n*] input for a dual-edge sample.

- 6x faster for Manchester coding, else the clock recovery block may function inappropriately.

47.3.7.4.3 Input clock configuration

To configure input clock edges and bitstream formats, you can:

- Write to C*n*CFR[ICSEL] to perform clock source selection and configure C*n*CFR[ICESEL] for the channel modulator clock to be rising edge, falling edge, or dual edge. These clock edges are used to sample the modulator bitstream.
- Write to C*n*CFR[IBFMT] for Manchester coding, when clocks are auto generated.

47.3.7.5 Clock source selection

The selected channel modulator clock input (EMCLK_SEL[*n*]) can be a dedicated external input (MCLK[*n*]), or it could be shared with an adjacent channel (MCLK[*n*-1]), if you write 111b to C*n*CFR[ICSEL]. Additionally, the shared clock can be cascaded between channels (for example, all channels share the same clock, MCLK[0]).

This is what you can do by writing appropriate values to the fields of Channel n Configuration (C0CFR - C4CFR):

- Route the output clocks OMCLK0, OMCLK1, and OMCLK2 as internal clock inputs by writing 000b, 001b, and 010b, respectively, to C*n*CFR[ICSEL]. Thus, SINC can provide output clocks to drive both the external modulator ADC and internal demodulator.
- Recover the manchester clock (MMCLK) by writing 01b to C*n*CFR[IBFMT]. You can use external modulator bitstream selection (EMBIT_SEL), which can be dedicated to each channel (external MBIT[*n*] and internal INP[*n*]) or shared with the adjacent channel (MBIT[*n*-1]), if you write 11b to C*n*CFR[IBSEL].
- Program the Manchester format by writing an appropriate value to C*n*CFR[ICESEL].

See Clock source selection settings for the values you must write to C*n*CFR[ICSEL], C*n*CFR[IBFMT], and C*n*CFR[ICESEL] for clock source selection.

47.3.7.5.1 Clock source selection settings

Table 412. Clock source selection settings

C*n*CFR[ICSEL]	C*n*CFR[IBFMT]	C*n*CFR[ICESEL]	Source	Description
000	00	Any value	OMCLK0	Output clock 0 selected
001	00	Any value	OMCLK1	Output clock 1 selected
010	00	Any value	OMCLK2	Output clock 2 selected
011	00	Any value	MCLK	External modulator clock selected
100	00	Any value	—	—
101	00	Any value	—	Reserved
110	00	Any value		
111	00	Any value	GCLK	Group clock (a previous channel clock selected as clock source)
Any value	01	01	MMCLK_RISE	A manchester clock selected per channel in the Rise format
		10	MMCLK_FALL	A manchester clock selected per channel in the Fall format
		00	Reserved	—
		11		

Table continues on the next page...

Table 412. Clock source selection settings (continued)

CnCFR[ICSEL]	CnCFR[IBFMT]	CnCFR[ICESEL]	Source	Description
Any value	10	Any value	PMCLK	An internal (IMCLK0), auto-generated parallel or serial modulator clock selected for clock source generation when you write a bitstream to Channel n Multipurpose Data (C0MPDATA - C4MPDATA)
	11		SMCLK	

47.3.7.6 Clock edge selection

For clock edge selection, you must program CnCFR[ICESEL] to select when to sample a modulator bitstream (EMBIT_SEL), if CnCFR[IBFMT] = 0.

See Clock edge selection settings for information about which modulator input clock (EMCLK_SEL) and clock edge you must select to sample this modulator bit (EMBIT_SEL), and see Clock edge selection diagram for a pictorial representation of these edge settings.

47.3.7.6.1 Clock edge selection settings

Table 413. Clock edge selection settings

CnCFR[IBFMT]	CnCFR[ICESEL]	Name	Clock edge used to sample the modulator bit (EMBIT_SEL)
00	001	MCLK positive edge	EMCLK_SEL as positive edge sample—SINC uses even-numbered MCLK edges (0, 2, 4, 6, 8, and so on) to sample the selected input data.
	010	MCLK negative edge	EMCLK_SEL as negative edge sample—SINC uses odd-numbered MCLK edges (1, 3, 5, 7, 9, and so on) to sample the selected input data.
	011	MCLK dual edge	EMCLK_SEL as dual edge sample—SINC uses all MCLK edges to sample the selected input data.
	101	MCLK positive edge even	EMCLK_SEL as every other even positive edge sample—SINC uses even-numbered MCLK edges (0, 4, 8, 12, 16, and so on) to sample the selected input data.
	110	MCLK negative edge odd	EMCLK_SEL as every other odd negative edge sample—SINC uses odd-numbered MCLK edges (1, 5, 9, 13, 17, and so on) to sample the selected input data.
	000—111	Reserved	—
01	00		
	01	MMCLK_RISE	A manchester clock per channel in the Rise format—no edge sample options.
	10	MMCLK_FALL	A manchester clock per channel in the Fall format—no edge sample options.
	11	Reserved	—

Table continues on the next page...

Table 413. Clock edge selection settings (continued)

CnCFR[IBFMT]	CnCFR[ICESEL]	Name	Clock edge used to sample the modulator bit (EMBIT_SEL)
10	Any value	PMCLK	An auto-generated internal clock when you write a bitstream to Channel n Multipurpose Data (C0MPDATA - C4MPDATA)—no edge sample options. IMCLK0 is selected for clock source generation.
11		SMCLK	

47.3.7.6.2 Clock edge selection diagram

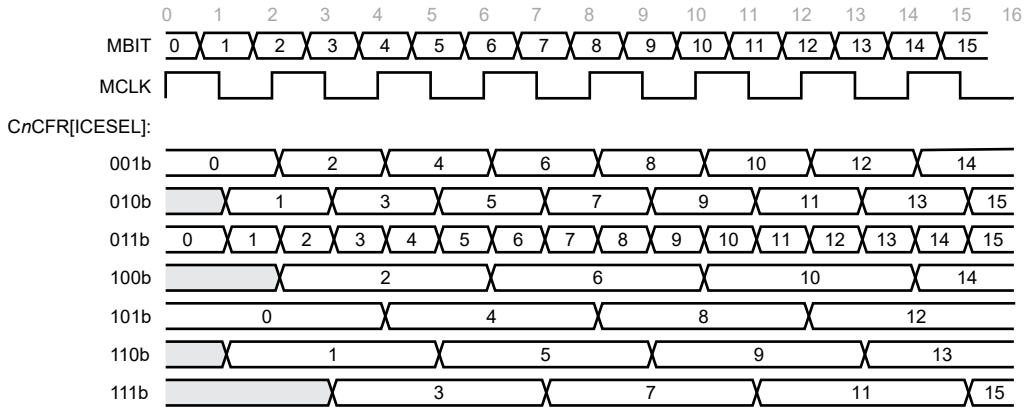


Figure 239. Clock edge selection diagram

47.3.7.7 Modulator clock frequencies and bit rates

The frequency of the function clock provided by the chip determines the minimum and maximum modulator clock frequencies.

47.3.7.7.1 Minimum clock output example

Assume the frequency of the function clock is 133 MHz. At this frequency, the minimum clock output frequency for MCLK_OUT2 is 16 kHz, and an internal or external clock can route it to SINC. Alternatively, an external ADC can divide the clock by 2. Therefore, in this example, the bit rate is 16 kbit/s or 8 bit/s in ideal scenarios.

See Minimum clock output settings for minimum clock output frequency settings of signals at different data rates and the resultant conditions.

47.3.7.7.1.1 Minimum clock output settings

Table 414. Minimum clock output settings

Signal	Minimum frequency	Condition	CnCFR[ICESEL]	Data rate (kbit/s)	Comments
MCLK_OUT0	65 kHz (133 MHz ÷ 8 ÷ 256 = 65 kHz)	Clock routed (CnCFR[ICSEL] = 000b)	001 = positive edge 010 = negative edge	65	—
			011	130	

Table continues on the next page...

Table 414. Minimum clock output settings (continued)

Signal	Minimum frequency	Condition	CnCFR[ICESEL]	Data rate (kbit/s)	Comments
			1xx	32	If ADC divides the clock by 2 but does not connect it
		ADC routes the clock after dividing it by 2 (CnCFR[ICSEL] = 011b)	001 = positive edge 010 = negative edge	32	—
			011	65	
			1xx	16	
MCLK_OUT1	32 kHz (133 MHz ÷ 8 ÷ 256 ÷ 2 = 32 kHz)	Clock routed (CnCFR[ICSEL] = 001b)	001 = positive edge 010 = negative edge	32	
			011	65	
			1xx	16	
		ADC routes the clock after dividing it by 2 (CnCFR[ICSEL] = 011b)	001 = positive edge 010 = negative edge	16	
			011	32	
			1xx	8	
MCLK_OUT2	16 kHz (133 MHz ÷ 8 ÷ 256 ÷ 2 ÷ 2 = 16 kHz)	Clock routed (CnCFR[ICSEL] = 010b)	001 = positive edge 010 = negative edge	16	
			011	32	
			1xx	8	
		ADC routes the clock after dividing it by 2 (CnCFR[ICSEL] = 011b)	001 = positive edge 010 = negative edge	8	
			011	16	
			1xx	4	

47.3.7.7.2 Maximum clock output example

Assume the frequency of the function clock is 133 MHz. At this frequency:

- 66 MHz is the maximum clock output frequency for MCLK_OUT0. The actual frequency depends on the chip I/O timing and the specific use case.

- 44 Mbit/s is the maximum data rate that SINC can support.

Therefore, an external ADC must divide a 66 MHz clock by 2 and route it as 33 MHz. The chip's I/O timing may also limit this high-speed clock.

With an output of 44 MHz, the duty cycle cannot be 50% when the clock is divided by 3. Most external ADCs require a 50% duty cycle. Therefore, 33 MHz is a better choice for the maximum frequency.

See Maximum clock output settings for maximum clock output frequency settings of signals at different data rates and the resultant conditions.

47.3.7.7.2.1 Maximum clock output settings

Table 415. Maximum clock output settings

Signal	Condition	CnCFR[ICESEL]	Maximum frequency (MHz)	Data rate (Mbit/s)	Comments
MCLK_OUT0	Clock routed (CnCFR[ICSEL] = 000b)	001 or 010	33	33	133 MHz ÷ 4 = 33 MHz (50% duty cycle)
			44	44	133 MHz ÷ 3 = 44 MHz (66% duty cycle, depending on the maximum data rate)
		011	22	44	133 MHz ÷ 6 = 22 MHz (depending on the maximum data rate)
		1xx	66	33	133 MHz ÷ 2 = 66 MHz
	ADC routes the clock after dividing it by 2 (CnCFR[ICSEL] = 011b)	001 or 010	66	33	133 MHz ÷ 2 = 66 MHz
		011	33	33	133 MHz ÷ 4 = 33 MHz (50% duty cycle)
			44	44	133 MHz ÷ 3 = 44 MHz (66% duty cycle, depending on the maximum data rate)
		1xx	66	17	133 MHz ÷ 2 = 66 MHz (SINC does not support clock division by 1)
MCLK_OUT1	Clock routed (CnCFR[ICSEL] = 001b)	001 or 010	17	17	133 MHz ÷ 4 ÷ 2 = 17 MHz (assuming you are also using MCLK_OUT0)
			33	33	133 MHz ÷ 2 ÷ 2 = 33 MHz (depending on the maximum output)
		011	17	33	133 MHz ÷ 4 ÷ 2 = 17 MHz (assuming you are also using MCLK_OUT0)
			22	44	133 MHz ÷ 3 ÷ 2 = 22 MHz (depending on the maximum output)
		1xx	17	8	133 MHz ÷ 4 ÷ 2 = 17 MHz (assuming you are also using MCLK_OUT0)
			33	17	133 MHz ÷ 2 ÷ 2 = 33 MHz (depending on the maximum output)
	ADC routes the clock after dividing it by 2 (CnCFR[ICSEL] = 011b)	001 or 010	17	8	133 MHz ÷ 4 ÷ 2 = 16 MHz (assuming you are also using MCLK_OUT0)
			33	17	133 MHz ÷ 2 ÷ 2 = 33 MHz (depending on the maximum output)

Table continues on the next page...

Table 415. Maximum clock output settings (continued)

Signal	Condition	CnCFR[ICESEL]	Maximum frequency (MHz)	Data rate (Mbit/s)	Comments
		011	17	17	133 MHz ÷ 4 ÷ 2 = 16 MHz (assuming you are also using MCLK_OUT0)
			33	33	133 MHz ÷ 2 ÷ 2 = 33 MHz (depending on the maximum output)
		1xx	17	4	133 MHz ÷ 4 ÷ 2 = 17 MHz (assuming you are also using MCLK_OUT0)
			33	8	133 MHz ÷ 2 ÷ 2 = 33 MHz (depending on the maximum output)
MCLK_OUT2	Clock routed (CnCFR[ICSEL] = 010b)	001 or 010	8	8	133 MHz ÷ 4 ÷ 2 ÷ 2 = 8 MHz (assuming you are also using MCLK_OUT0)
			17	17	133 MHz ÷ 2 ÷ 2 ÷ 2 = 17 MHz (depending on the maximum output)
		011	8	17	133 MHz ÷ 4 ÷ 2 ÷ 2 = 8 MHz (assuming you are also using MCLK_OUT0)
			17	33	133 MHz ÷ 2 ÷ 2 ÷ 2 = 17 MHz (depending on the maximum output)
		1xx	8	4	133 MHz ÷ 4 ÷ 2 ÷ 2 = 8 MHz (assuming you are also using MCLK_OUT0)
			17	8	133 MHz ÷ 2 ÷ 2 ÷ 2 = 17 MHz (depending on the maximum output)
	ADC routes the clock after dividing it by 2 (CnCFR[ICSEL] = 011b)	001 or 010	8	4	133 MHz ÷ 4 ÷ 2 ÷ 2 = 8 MHz (assuming you are also using MCLK_OUT0)
			17	8	133 MHz ÷ 2 ÷ 2 ÷ 2 = 17 MHz (depending on the maximum output)
		011	8	8	133 MHz ÷ 4 ÷ 2 ÷ 2 = 8 MHz (assuming you are also using MCLK_OUT0)
			17	17	133 MHz ÷ 2 ÷ 2 ÷ 2 = 17 MHz (depending on the maximum output)
		1xx	8	2	133 MHz ÷ 4 ÷ 2 ÷ 2 = 8 MHz (assuming you are also using MCLK_OUT0)
			17	4	133 MHz ÷ 2 ÷ 2 ÷ 2 = 17 MHz (depending on the maximum output)

47.3.7.7.3 Maximum clock input example

The modulator input clock can be either routed from a SINC output clock or provided by an external ADC (see Block diagram for details).

The maximum input frequency allowed depends on:

- The maximum input data rate and use case.
- The chip's I/O timing.

Assuming that the frequency of the Function clock is 133 MHz, the maximum data rate is 44 Mbit/s (133MHz ÷ 3 = 44 Mbit/s) for a typical use case. For Manchester formats, the maximum allowed clock frequency (recovery from bit) is 22 MHz (133 MHz ÷ 6 = 22 Mbit/s).

The minimum allowed input frequency is not limited if the clock input comes from an external ADC. However, if you select the Manchester format, the limit is 133 MHz ÷ 1024 ÷ 8 = 16 kHz.

NOTE

To calculate the output sample rate, ORD = 3 in Continuous mode.

See Maximum clock input settings for maximum clock input frequency settings of signals at different data rates and the resultant conditions.

47.3.7.7.3.1 Maximum clock input settings

Table 416. Maximum clock input settings

Case	CnCFR[ICESEL]	Maximum frequency (MHz)	Input data rate (Mbit/s)	OSR	Output sample rate	Comments
CnCFR[IBFMT] = 00	001 010	44	44	16	2.7 Mbit/s	44 MHz ÷ 16 = 2.7 Mbit/s; 14-bit CIC
				256	172 kbit/s	44 MHz ÷ 256 = 172 kbit/s; 26-bit CIC
				1290	34 kbit/s	44 MHz ÷ 1290 = 34 kbit/s; 32-bit CIC
	011	22	44	16	2.7 Mbit/s	—
				256	172 kbit/s	
				1290	34 kbit/s	
	1xx	44	22	16	1.4 Mbit/s	
				256	86 kbit/s	
				1290	17 kbit/s	
CnCFR[IBFMT] = 01	Any value	22	22	16	1.4 Mbit/s	For the Manchester format, recovered clocks must be a value smaller than 22 Mbit/s.
				256	86 kbit/s	—
				1290	17 kbit/s	

47.3.7.8 Modulator clock I/O timing

47.3.7.8.1 I/O timing diagram

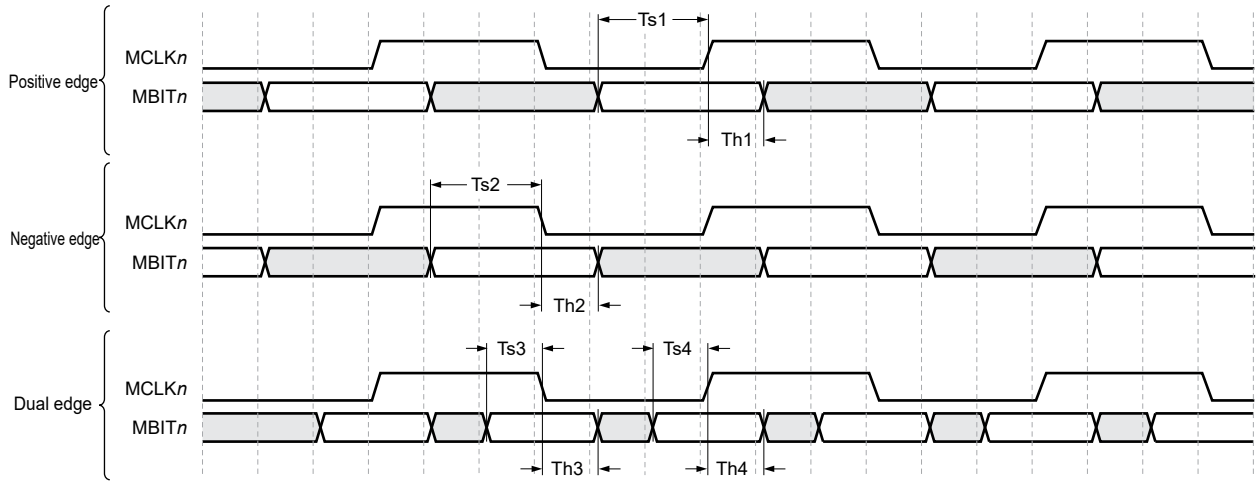


Figure 240. I/O timing diagram

See I/O timing settings for the symbols used in this diagram.

47.3.7.8.2 I/O timing settings

Table 417. I/O timing settings

Symbol	Description	Minimum time (in ns)	Maximum time (in ns)
T_{s1}	Setup time from data valid to clock high	5	—
T_{h1}	Hold time from clock high to data valid	5	—
T_{s2}	Setup time from data valid to clock low	5	—
T_{h2}	Hold time from clock low to data valid	5	—
T_{s3}	Setup time from data valid to clock high	5	—
T_{h3}	Hold time from clock high to data valid	5	—
T_{s4}	Setup time from data valid to clock low	5	—
T_{h4}	Hold time from clock low to data valid	5	—

47.3.8 Reset

This section describes how to reset the SINC module and explains special requirements related to reset.

Writing 1 to MCR[RST] resets the following function blocks and registers:

- All function blocks (such as filters, short-circuit detectors, and so on) except for the clock
- Normal Interrupt Status (NIS)
- Error Interrupt Status (EIS)
- FIFO And CAD Error Interrupt Status (FIFOIS)
- Status (SR)

- MCR[STRIG*n*], together with related functions

47.3.9 Interrupts and breaks

This section describes all the interrupts that SINC generates.

Each channel has several interrupt sources that you can control and check with dedicated register fields.

If NIE[COCIE*n*] = 1, then an interrupt request is used to indicate that one filtering result is available in the corresponding channel's FIFO. You can then read the result in Channel n Result Data (C0RDATA - C4RDATA).

If NIE[CHFIE*n*] = 1, then an interrupt request is used to indicate that the FIFO surpassed its watermark level. In this case, SINC sets the corresponding NIS[CHF*n*] flag.

SINC can also generate an error interrupt request when an unexpected condition such as a FIFO overflow or underflow, a loss of clock, saturation, or exceeded value threshold occurs.

When an error occurs, SINC can assert break signals to deactivate an external peripheral. You control this feature by using Channel n Protection (C0PROT - C4PROT).

Table 418. Interrupts and breaks

Status register	Status field	Enable register	Enable field	Break enable field in C*n*PROT	Description
NIS	COC*n*	NIE	COCIE*n*	—	Conversion complete
	CHF*n*		CHIE*n*	—	Data ready
	ZCS*n*		ZCIE*n*	—	Zero cross detected
EIS	SCD*n*	EIE	SCDIE*n*	SCDBK	Short circuit detected
	WLMT*n*		WLMTIE*n*	WLMTBK	Window limit reached
	LLMT*n*		LLMTIE*n*	LLMTBK	Low limit reached
	HLMT*n*		HLMTIE*n*	HLMKBK	High limit reached
FIFOIS	FUNF*n*	FIFOIE	FUNFIE*n*	—	FIFO underflow
	FOVF*n*		FOVFIE*n*	—	FIFO overflow
	CADF*n*		CADIE*n*	CADBK	Clock absence detected
	SAT*n*		SATIE*n*	—	Saturation

47.3.10 DMA

SINC has two DMAs for each channel:

- Primary DMA exclusively for the FIFO
- Alternate DMA for various sources (see C*n*ACFR[ADMASEL])

If C*n*CR[DMAEN] = 1, you read the FIFO-stored samples through DMA transactions. Each channel has dedicated DMA requests. If the FIFO of each channel reach their watermark the output interface makes a request for DMA transaction.

SINC clears the "data output ready" flags (NIS.CHFn) when a DMA transaction finishes.

SINC supports all the alternate DMA sources mentioned in C*n*ACFR[ADMASEL]. You can select only one source at a time. After an alternate DMA transaction finishes, SINC clears the related flags in Normal Interrupt Status (NIS), Error Interrupt Status (EIS), and FIFO And CAD Error Interrupt Status (FIFOIS).

47.4 External signals

Table 419. External signals

Signal	Used for	Description
BREAK_CAD[4:0]	Output to the chip	CAD-related break signals sent to the chip for turning off other functions
BREAK_HIGH[4:0]	Output to the chip	High-limit break signals sent to the chip for turning off other functions
BREAK_LOW[4:0]	Output to the chip	Low-limit break signal sent to the chip for turning off other functions
BREAK_SCD[4:0]	Output to the chip	SCD-related break signal sent to the chip for turning off other functions
BREAK_WIN[4:0]	Output to the chip	Window-limit-related break signal sent to the chip for turning off other functions
FILT_CC[4:0]	Output to the chip	Signal acting as an output to be used by downstream modules in the signal processing chain that SINC is a component of
HTRIG[4:0]	Input from the chip	Hardware external trigger
INP[4:0]	Input from the chip	Internal modulator bitstream input
MBIT[4:0]	Input from the chip	External modulator bitstream input
MCLK_OUT0	Output to the chip	Modulator clock output 0
MCLK_OUT1	Output to the chip	Modulator clock output 1, which is MCLK_OUT0 divided by 2
MCLK_OUT2	Output to the chip	Modulator clock output 2, which is MCLK_OUT1 divided by 2
MCLK[4:0]	Input from the chip	External modulator clock input
ipi_coc_int[3:0]	Output to the chip	Conversion complete interrupt
ipi_chf_int[3:0]	Output to the chip	Channel (FIFO) output interrupt
ipi_zcs_int[3:0]	Output to the chip	Zero cross interrupt
ipi_scd_int[3:0]	Output to the chip	Short circuit detect interrupt
ipi_low_int[3:0]	Output to the chip	Low limit interrupt
ipi_high_int[3:0]	Output to the chip	High limit interrupt
ipi_win_int[3:0]	Output to the chip	Window limit interrupt
ipi_fund_int[3:0]	Output to the chip	FIFO underflow interrupt
ipi_fover_int[3:0]	Output to the chip	FIFO overflow interrupt
ipi_sat_int[3:0]	Output to the chip	Saturation interrupt
ipi_cad_int[3:0]	Output to the chip	Clock monitor interrupt
ipi_chn_ored_int[3:0]	Output to the chip	ORed together for above interrupt per channel

Table continues on the next page...

Table 419. External signals (continued)

Signal	Used for	Description
ipd_req_sinc[3:0]	Output to the chip	Conversion complete DMA request for SINC
ipd_done_sinc[3:0]	Input from the chip	Handshake signal from chip, indicates that DMA request is processed
ipd_req_alt[3:0]	Output to the chip	Alternative DMA request for SINC
ipd_done_alt[3:0]	Input from the chip	Handshake signal from chip, indicates that DMA request is processed

47.5 Initialization

47.5.1 Starting SINC

By default, SINC is disabled (MCR[MEN] = 0). To start or configure SINC, do this:

1. Configure the shared clock generation logic as described in Configure clocking-related fields before enabling SINC.
2. Configure OSR, ORD, and all other necessary options.
3. Enable the modulator clock output.
4. Enable all the required decimation filters and protection functions.
5. Write 1 to MCR[MEN] to start SINC.

The corresponding clock is generated for both the internal (IMCLK0) and external (MCLK_OUT*n*) signals (see External signals). If MCLK_OUT*n* is enabled, SINC waits for seven clock cycles before the external ADC is stable. SR[MCLKRDY*n*] fields become 1 after that.

This is how SINC behaves during and after starting up:

- For an internal clock source (if C*n*CFR[ICSEL] = 000b, 010b, or 001b), MCLK_OUT*n* checks SR[MCLKRDY*n*]. The field changes into a valid state when it becomes 1.
- If the clock source is external, Manchester, or grouped, all the three SR[MCLKRDY*n*] fields are selected and wait for a certain number of clock cycles before becoming 1.
- If MCLK_OUT*n* is disabled, SINC waits for one clock cycle for the internal logic to be stable.
- After startup, SINC waits for the conversion trigger to start conversion. The enabled channel conversions begin after SINC receives a software or hardware trigger. During startup, SINC ignores this signal.

47.5.2 Configure clocking-related fields before enabling SINC

Before you enable SINC (write 1 to MCR[MEN]), you must configure the following fields:

Table 420. Configure clocking-related fields before enabling SINC

Register	Fields to configure
Main Control (MCR)	MCLK0DIS MCLK1DIS MCLK2DIS MCLKDIV PRESCALE

47.5.3 Configure fields before enabling channels

Before you enable a channel (write 1 to C*n*CR[CHEN]), you must configure the following fields:

Table 421. Configure fields before enabling channels

Register	Fields to configure
Channel n Data Rate (C0DR - C4DR)	All
Channel n Configuration (C0CFR - C4CFR)	All except FIFOWMK
Channel n Protection (C0PROT - C4PROT)	All
Channel n Bias (C0BIAS - C4BIAS)	All
Channel n Low Limit (C0LOLMT - C4LOLMT)	All
Channel n High Limit (C0HILMT - C4HILMT)	All
Channel n Advanced Configuration (C0ACFR - C4ACFR)	All

47.6 Application information

This section describes several ways to use SINC.

47.6.1 Reading asynchronous data safely

1. Write 1 to C*n*SR[SRDS].
2. Wait until SINC changes C*n*SR[SRDS] to 0.
3. Read the data in Channel n Debug (C0DBGR - C4DBGR).

47.6.2 Transferring data between domains safely

To transfer data from the function-clock domain to the bus-clock domain safely, perform this procedure:

1. Choose the type of data that you want (see Details of debug data).
2. Write the corresponding value to C*n*CR[DBGSEL].
3. Write 1 to C*n*SR[SRDS].
4. Wait until C*n*SR[SRDS] becomes 0.
5. Read the data from C*n*DBGR.
6. Ensure that C*n*SR[DBGRS] = 0.

If you skip steps 1 and 2, C*n*DBGR contains the data from the last time you performed the full procedure.

Due to SINC's internal microarchitecture, the returned instantaneous data may be incomplete and invalid. Step 6 ensures that the data you read is valid.

47.6.3 Use case: Motor control with Single mode

47.6.3.1 Overview

This example uses Single mode to control a typical motor with three coils. Six on-chip PWM channels drive those three coils. Three external ADCs measure the coil current, and additional ADCs monitor total current or voltage to trigger critical events (such as overcurrent or short circuit) to shut down the motor.

The ADCs produce a high-speed bitstream and send it to SINC through MBIT signals (see External signals). Then, SINC:

1. Demodulates the bitstream.

2. Decimates the bitstream.
3. Converts the bitstream to a 24-bit result at a lower frequency.
4. Saves the result in a FIFO.

You can then:

1. Read the coil current from the FIFO.
2. Read other information (such as speed and position) from other modules on the chip.
3. Process the data to determine the parameters (on or off, PWM ratio drive strength) for the next coil-drive step.

Schematic shows the motor-control schematic.

Frequency response shows the resulting frequency response of the PF's CIC. In that diagram, pay particular attention to the following:

- Without a compensation filter, there is a drop in the passband, and roll-off (aliasing) occurs in the stop band.
- F_D (346 kHz) is near the 17th harmonic wave of the PWM switch frequency (20 kHz). This example filters out this noise more efficiently.
- A dropoff of –3 dB occurs at F_C (91 kHz). Therefore, the input signal must have a much-lower frequency—1 kHz or lower. If this is not possible, your software must compensate accordingly.

47.6.3.2 Schematic

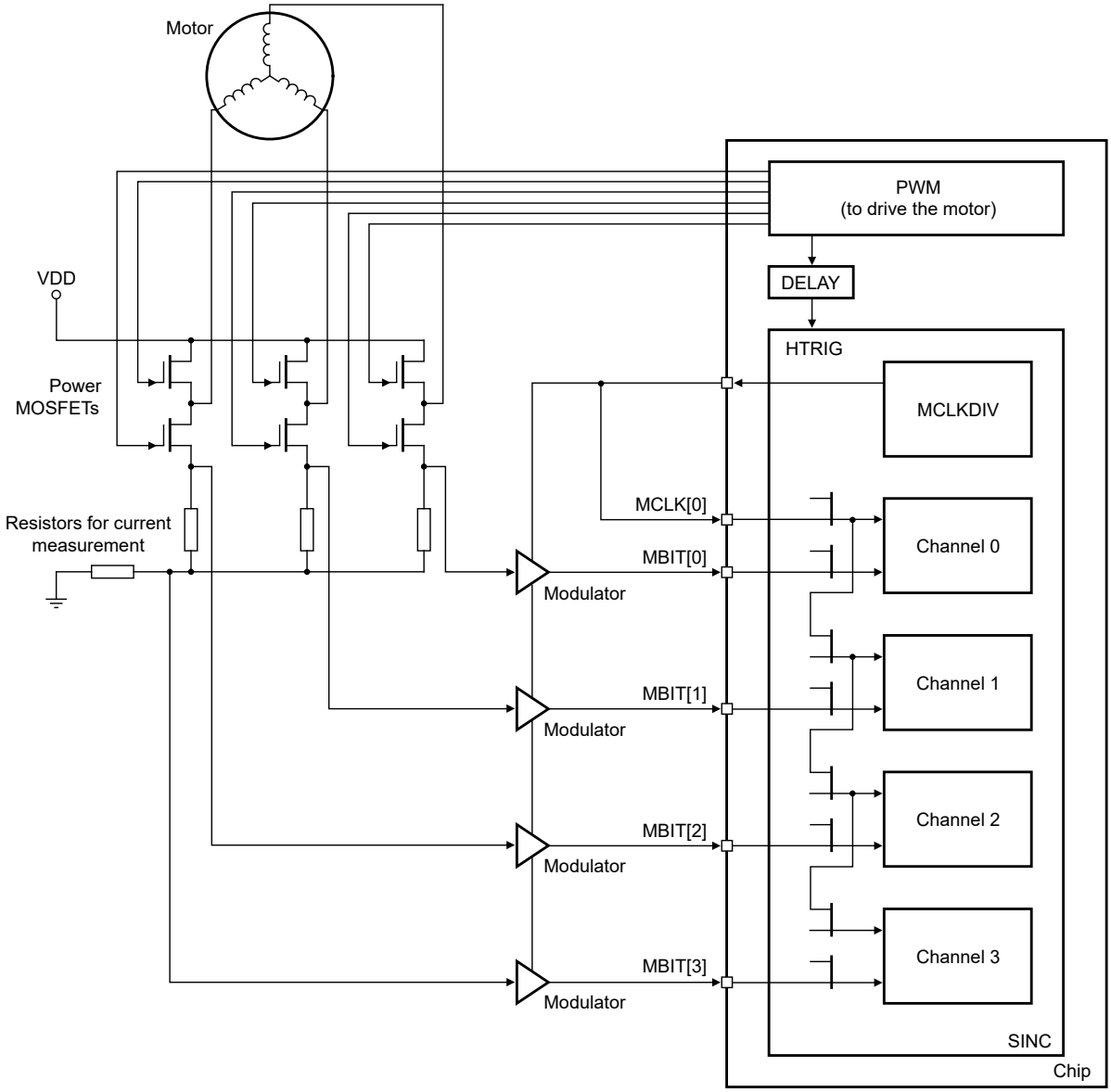


Figure 241. Schematic

47.6.3.3 Assumptions

Table 422. Assumptions

Parameter	Value
SINC function clock	133 MHz
Target external modulator ADC clock (F_S)	22.2 MHz
Motor speed	6000 RPM (100 Hz)

Table continues on the next page...

Table 422. Assumptions (continued)

Parameter	Value
PWM switch frequency (F_P)	20 kHz (corresponding to a 50 µs period, or 200 PWM steps per revolution)
OSR	64
Decimation frequency (F_D)	346 kHz (calculated from target external modulator ADC clock and OSR)
Cutoff frequency (F_C)	91 kHz
Signal-to-noise ratio (SNR)	79 dB
Effective number of bits (ENOB)	Approximately 12.9 (calculated using SNR)
Data width	20 bits (calculated using OSR and ORD)

47.6.3.4 Frequency response

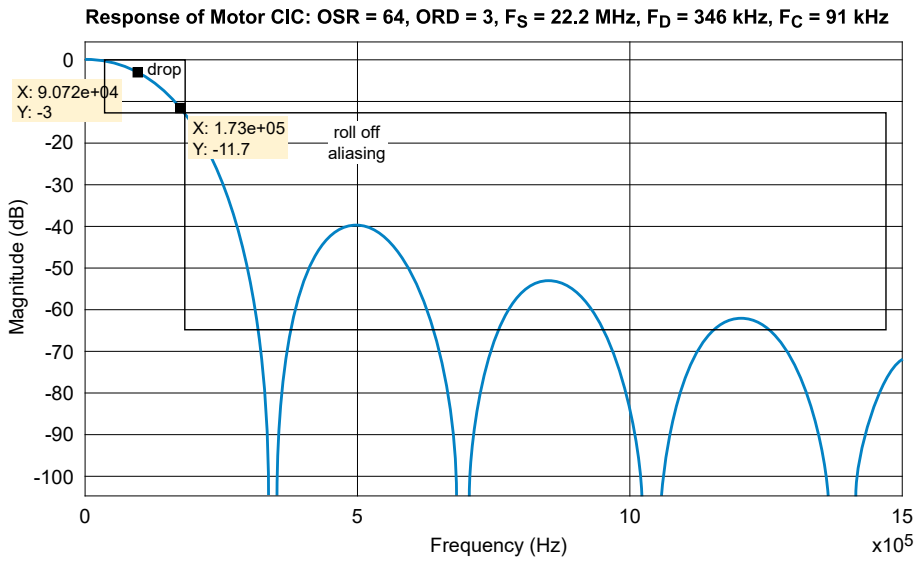


Figure 242. Frequency response

47.6.3.5 Procedure

1. Configure the related PWM and other chip modules to establish the correct pin connection with SINC. In particular, ensure that the PWM event signal connects to SINC as a hardware trigger, so that when PWM starts driving the coil, SINC can start conversion at a specific time.
2. Program the PWM event delay such that when SINC starts conversion, the coil currents are stable, as shown in Measurement time.
3. If the external ADC does not provide a clock, configure the modulator clock output. In this case, MCLK_OUT0 drives all external ADCs. To achieve MCLK_OUT0 = 22.2 MHz:
    a. Write 0 to MCR[PRESCALE].
    b. Write 5d to MCR[MCLKDIV].
4. Configure the data rate for channels 0–2 (*n* = 0–2) as follows:
    a. Write 3d to C*n*DR[PFORD].
    b. Write 63d to C*n*DR[PFOSR] to select OSR = 64.

c. Write 0 to C*n*DR[PFCM] to select Single mode.

5. Configure the data rate for channel 3 (*n* = 3) as follows:
   a. Write 2d to C*n*DR[PFORD] to select the optional fast response.
   b. Write 63d to C*n*DR[PFOSR] to select OSR = 64.
   c. Write 10b to C*n*DR[PFCM] to select Always mode for monitoring total current.
6. Configure fields in Channel n Configuration (C0CFR - C4CFR) for data operation as follows:

Channels (value of *n*)	C*n*CFR field to write	Value to write	Purpose
0–3	ITLVL	0	Select edge triggering
0–2	ITSEL	1	Select hardware triggering
3	ITSEL	0	Select software triggering
0–3	IBSEL IBFMT	0	Select a bitstream from the MBIT signals (see External signals)
0	ICSEL	11b	Select the external modulator clock
1–3	ICSEL	111b	Group the clocks with channel 0
0–3	ICESEL	1b or 10b	Configure the clock edge that you want
0–3	FIFOWMK	0	Set the FIFO watermark
0–3	RDFMT	0	Select the signed data format
0–3	PFSFT	1_0100b	Select a left shift of 4

NOTE

For channels 0–2, the conversion time (calculated using ORD×OSR÷F_S) is 8.6 µs. This is smaller than the PWM period (50 µs) and allows enough margin to adjust sample time that Measurement time requires.

7. Enable SINC features as follows:

Channels (value of *n*)	Register and field to write	Value to write	Purpose
0–3	C*n*CR[PFEN]	1	Enable the PFs
0–3	C*n*CR[CHEN]	1	Enable the channels and their subfunctions
0–3	C*n*CR[FIFOEN]	1	Enable each PF's FIFO transfers
3	C*n*CR[SCDEN]	1	Enable SCD
3	C*n*PROT[SCDOP]	0	Specify that both repeating bit values increment the SCD counter
3	C*n*PROT[SCDBK]	1	Enable the SCD break signal
3	C*n*PROT[SCDLMT]	192d	Specify the SCD limit threshold

8. Optionally enable other subfunctions if you need them.
9. To enable interrupts for all channels (*n* = 0–3), write 1 to NIE[COCIEn].
10. To enable conversion, write 1 to MCR[MEN].

11. Wait until SR[CHRDYn] becomes 1 for all channels (*n* = 0–3).
12. Start PWM.
13. To start conversion for channel 3, write 1 to MCR[STRIG3].
14. Wait until NIS[COC*n*] = 1. This indicates conversion completion.
15. Read the ADC results from the FIFO using Channel n Result Data (C0RDATA - C4RDATA).
16. Process the motor-control data.
17. Reconfigure PWM.
18. Go to step 14 for the next loop.

47.6.3.6 Measurement time

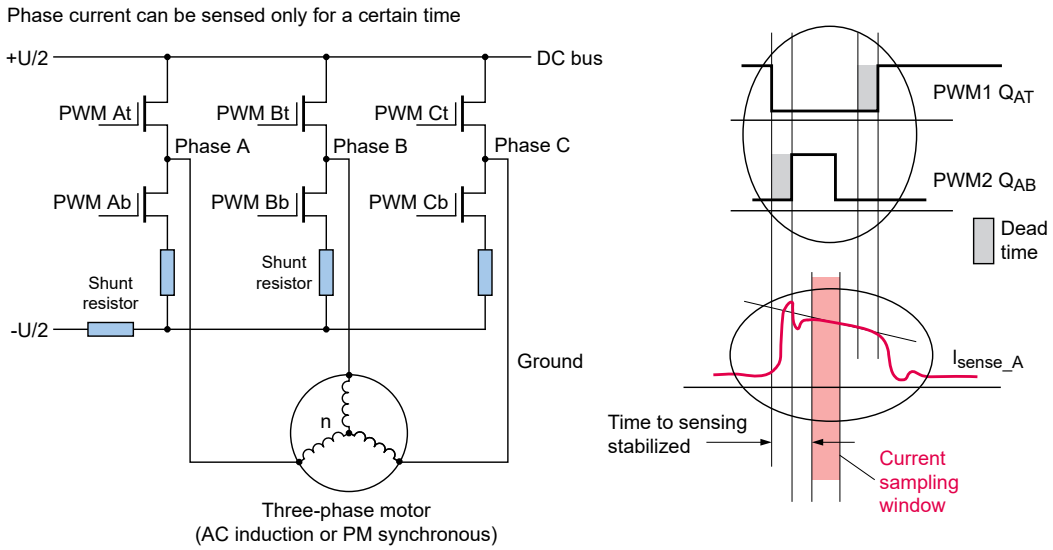


Figure 243. Measurement time

47.6.4 Use case: Motor control with Fixed-Number mode

47.6.4.1 Overview

This example improves on Use case: Motor control with Single mode by adding a compensation filter to achieve better precision, flatten the passband, improve roll-off, and so on. This method requires several samples for each PWM turn to perform filter compensation and sharpening with additional decimation. After the FIFO reaches its watermark, the conversion and filtering processes automatically stop.

NOTE

You can also achieve the same result with Continuous mode. When the FIFO reaches its watermark, you can toggle C*n*CR[PFEN] to abort conversion and wait for another trigger. The problem with this approach is that you must toggle C*n*CR[PFEN] as quickly as possible when SINC sets the corresponding NIS[CHF*n*] flag. If you disable conversion too late, it is hard to identify whether the 13th conversion had completed before you terminated conversion.

47.6.4.2 Assumptions

Table 423. Assumptions

Parameter	Value
SINC function clock	133 MHz
Target external modulator ADC clock (F_S)	22.2 MHz
Motor speed	6000 RPM (100 Hz)
PWM switch frequency (F_P)	20 kHz (corresponding to a 50 µs period, or 200 PWM steps per revolution)
Number of samples per PWM period	12
OSR	64
Decimation frequency (F_D)	346 kHz (calculated using the target external modulator ADC clock and OSR)
Cutoff frequency (F_C)	91 kHz
Additional decimation factor	2, in software
SNR	Approximately 94 dB (see the ADC specifications in the chip data sheet)
ENOB	Approximately 15.4 (calculated using SNR)
Data width	20 bits (calculated using OSR and ORD)

47.6.4.3 Frequency response

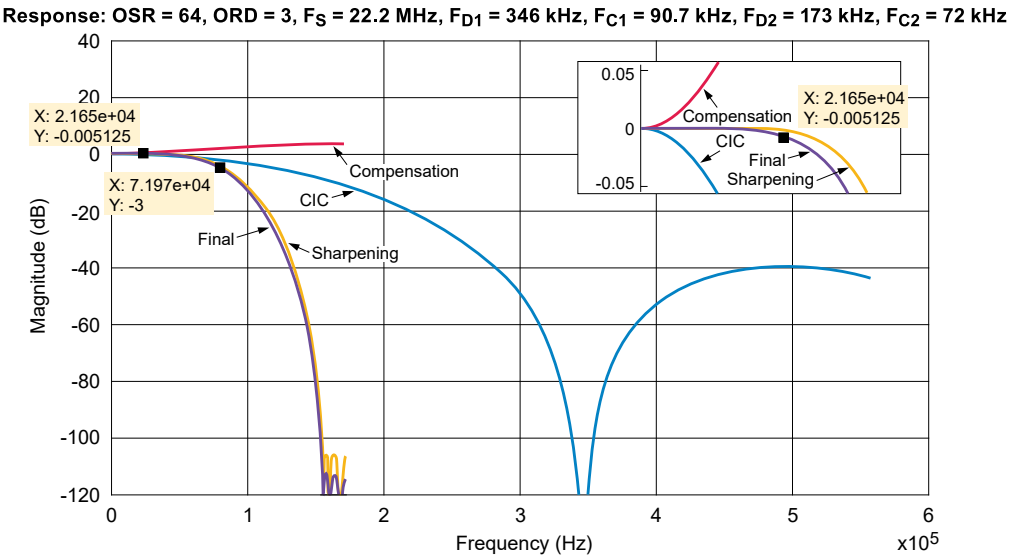


Figure 244. Frequency response

47.6.4.4 Procedure

1. Configure the related PWM and other chip modules to establish the correct pin connection with SINC. In particular, ensure that the PWM event signal connects to SINC as a hardware trigger, so that when PWM starts driving the coil, SINC can start conversion at a specific time.

2. Program the PWM event delay such that when SINC starts conversion, the coil currents are stable, as shown in Measurement time.
3. If the external ADC does not provide a clock, configure the modulator clock output. For this example, MCLK_OUT0 drives all external ADCs. To achieve MCLK_OUT0 = 22.2 MHz:
    a. Write 0 to MCR[PRESCALE].
    b. Write 5d to MCR[MCLKDIV].
4. Configure the data rate for channels 0–2 (*n* = 0–2) as follows:
    a. Write 3d to C*n*DR[PFORD].
    b. Write 63d to C*n*DR[PFOSR] to select OSR = 64.
    c. Write 11b to C*n*DR[PFCM] to select Fixed-Number mode.
5. Configure the data rate for channel 3 (*n* = 3) as follows:
    a. Write 2d to C*n*DR[PFORD] to select the optional fast response.
    b. Write 63d to C*n*DR[PFOSR] to select OSR = 64.
    c. Write 10b to C*n*DR[PFCM] to select Always On mode for monitoring total current.
6. Configure fields in Channel n Configuration (C0CFR - C4CFR) for data operation as follows:

Channels (value of *n*)	C*n*CFR field to write	Value to write	Purpose
0–3	ITLVL	0	Select edge triggering
0–2	ITSEL	1	Select hardware triggering
3	ITSEL	0	Select software triggering
0–3	IBSEL IBFMT	0	Select a bitstream from the MBIT signals (see External signals)
0	ICSEL	11b	Select the external modulator clock
1–3	ICSEL	111b	Group the clocks with channel 0
0–3	ICESEL	1b or 10b	Configure the clock edge that you want
0–3	FIFOWMK	11b	Set the FIFO watermark
0–3	RDFMT	0	Select the signed data format
0–3	PFSFT	1_0100b	Select a left shift of 4

NOTE

For channels 0–2, the conversion time for 12 samples (calculated using {[ORD+11]×OSR]}÷F_S) is 40.3 µs. This is smaller than the PWM period (50 µs) and allows enough margin to adjust sample time that Measurement time requires.

7. Enable SINC features as follows:

Channels (value of *n*)	Register and field to write	Value to write	Purpose
0–3	C*n*CR[PFEN]	1	Enable the PFs

Table continues on the next page...

Table continued from the previous page...

Channels (value of *n*)	Register and field to write	Value to write	Purpose
0–3	C*n*CR[CHEN]	1	Enable the channels and their subfunctions
0–3	C*n*CR[FIFOEN]	1	Enable each PF's FIFO transfers
3	C*n*CR[SCDEN]	1	Enable SCD
3	C*n*PROT[SCDOP]	0	Specify that both repeating bit values increment the SCD counter
3	C*n*PROT[SCDBK]	1	Enable the SCD break signal
3	C*n*PROT[SCDLMT]	192d	Specify the SCD limit threshold

8. Optionally enable other subfunctions if you need them.
9. To enable interrupts for all channels (*n* = 0–3), write 1 to NIE[COCIEn].
10. To enable conversion, write 1 to MCR[MEN].
11. Wait until SR[CHRDYn] becomes 1 for all channels (*n* = 0–3).
12. Start PWM.
13. To start conversion for channel 3, write 1 to MCR[STRIG3].
14. Wait until NIS[CHF*n*] = 1. This indicates conversion completion.
15. Read the ADC results from the FIFO using Channel n Result Data (C0RDATA - C4RDATA).
16. Optionally perform compensation:
    a. CIC gain adjust (divide 40_0000h if necessary) to get a fixed-point decimal between -1.0 and 1.0, than take part in the ADC reference voltage to get the final value in the correct format.
    b. Post-process the ADC result at a low speed, perform filter compensation and sharpening, and decimate the frequency by 2 to get the final ADC result as shown in Frequency response.
    c. If you have enough samples and don't have a hardware-delay feature, optionally perform delay compensation (for example, with a Lagrange polynomial).
17. Process the motor-control data.
18. Reconfigure PWM.
19. Go to step 14 for the next loop.

47.6.5 Use case: Audio system with Continuous mode

47.6.5.1 Overview

You can use SINC for two typical audio applications:

- Stereo recorder, using two SINC channels to sample the left and right audio channels
- Microphone array, using two or more SINC channels to sample a person's voice (based on distance from the person) and a beamforming software compensation to filter out environment noise

Schematic shows the audio schematic.

Waveforms: Audio system illustrates why you need different edges and why one MBIT input goes to two channels.

Frequency response after CIC filter shows the resulting frequency response of the PF's CIC.

Frequency response after HPF shows that HPF response is not linear and does not approach 0 dB.

47.6.5.2 Schematic

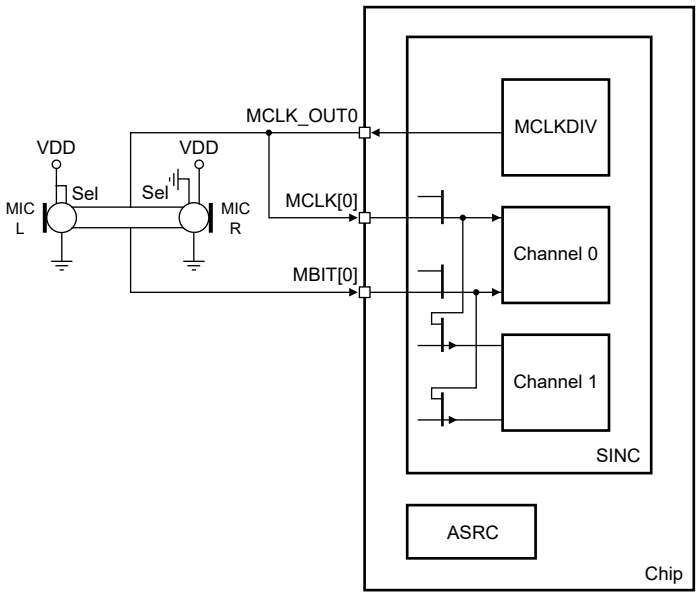


Figure 245. Schematic

47.6.5.3 Assumptions

Table 424. Assumptions

Parameter	Value
SINC function clock	133 MHz
Target external modulator ADC clock	16.6 MHz
Target audio sample rate	44.0 kHz
Data precision	24 bits
OSR	94
Additional decimation factor	4
C*n*ACFR[HPFA]	1
Signal-to-noise ratio	Approximately 118 dB
Equivalent number of bits	Approximately 19.2
Final output frequency	44.22 kHz (calculated using the ADC clock, OSR, and additional decimation)

47.6.5.4 Waveforms: Audio system

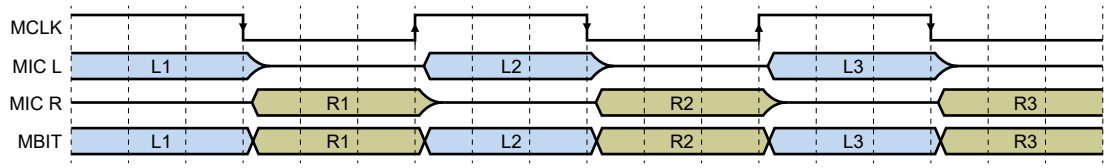


Figure 246. Waveforms: Audio system

47.6.5.5 Frequency response after CIC filter

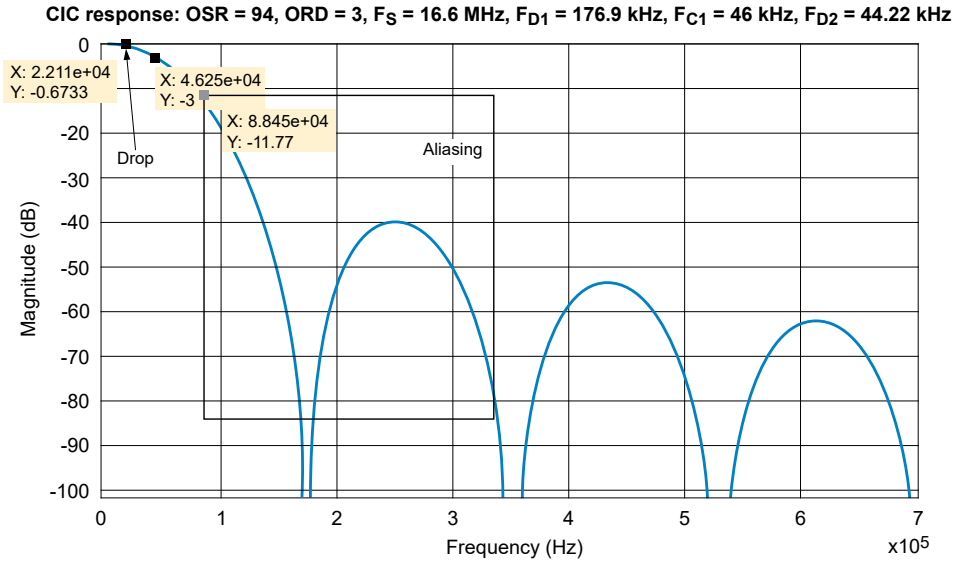


Figure 247. Frequency response after CIC filter

47.6.5.6 Frequency response after LPF and compensation

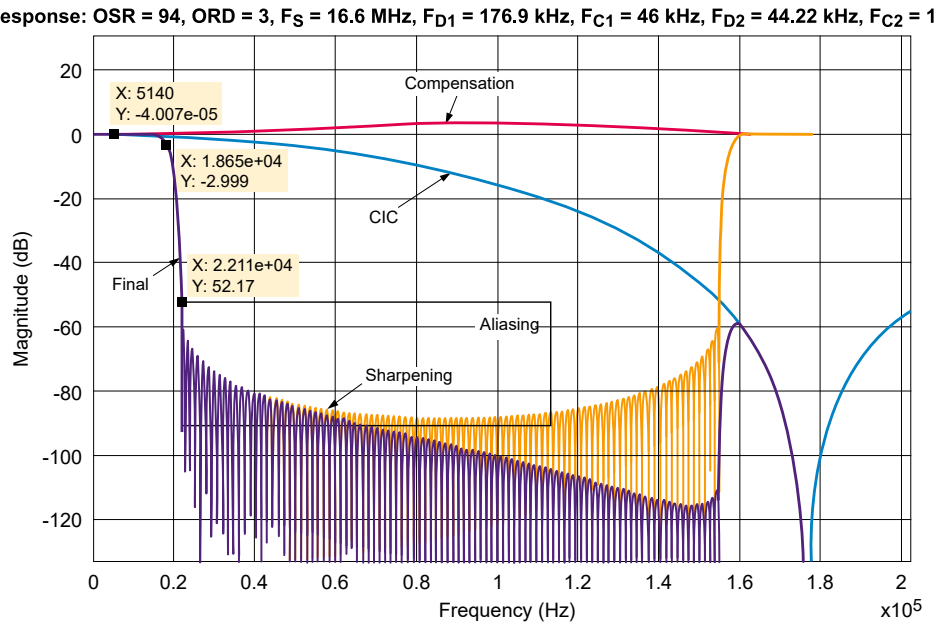


Figure 248. Frequency response after LPF and compensation

47.6.5.7 Frequency response after HPF

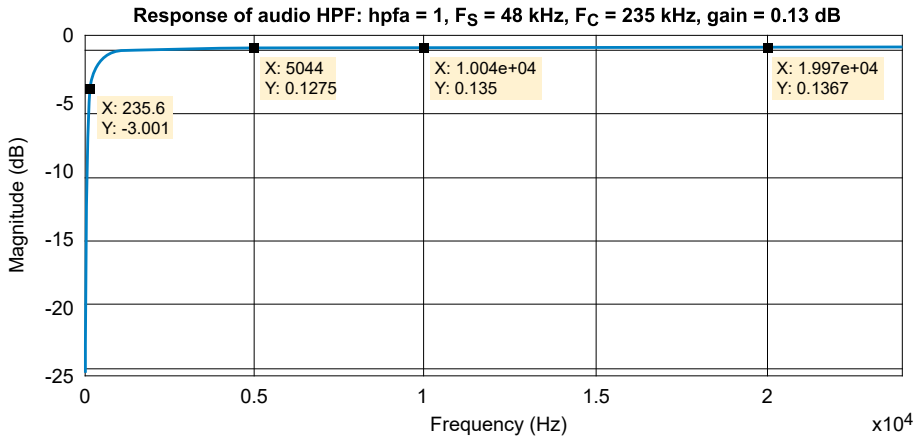


Figure 249. Frequency response after HPF

47.6.5.8 Procedure

This procedure involves only channels 0 and 1 (*n* = 0–1).

1. Configure the chip pin connections so that one MBIT goes to both SINC channels 0 and 1 (containing the left and right bitstreams, respectively).
2. If the external ADC does not provide a clock, configure the modulator clock output. For this use case, MCLK_OUT0 drives all external ADCs. To achieve MCLK_OUT0 = 16.6 MHz:
   a. Write 0 to MCR[PRESCALE].
   b. Write 7d to MCR[MCLKDIV].

3. Configure the data rate as follows:
    a. Write 3d to C*n*DR[PFORD].
    b. Write 93d to C*n*DR[PFOSR] to select OSR = 94.
    c. Write 01b to C*n*DR[PFCM] to select Continuous mode.
4. Configure fields in Channel n Configuration (C0CFR - C4CFR) for data operation as follows:

Channels (value of *n*)	C*n*CFR field to write	Value to write	Purpose
0–1	ITLVL	0	Select edge triggering
0–1	ITSEL	0	Select software triggering
0–1	IBSEL IBFMT	0	Select a bitstream from the MBIT signals (see External signals)
0	ICSEL	11b	Select the external modulator clock
1	ICSEL	111b	Group the clocks with channel 0
0	ICESEL	1	Select the left clock edge
1	ICESEL	10b	Select the right clock edge
0–1	FIFOWMK	7d	Set the FIFO watermark
0–1	RDFMT	0	Select the signed data format
0–3	PFSFT	1_0011b	Select a left shift of 3

5. Enable SINC features:
    - Optionally enable other subfunctions if you need them.

Channels (value of *n*)	Register and field to write	Value to write	Purpose
0–1	C*n*CR[PFEN]	1	Enable the PFs
0–1	C*n*CR[CHEN]	1	Enable the channels and their subfunctions
0–1	C*n*CR[FIFOEN]	1	Enable each PF's FIFO transfers
0–1	C*n*ACFR[HPFA]	1	Select the alpha coefficient

6. Optionally enable other subfunctions if you need them.
7. To enable interrupts for channels 0 and 1, write 1 to CHFIE0 and CHFIE1 in Normal Interrupt Enable (NIE).
8. To enable conversion, write 1 to MCR[MEN].
9. Wait until SR[CHRDYn] becomes 1.
10. To start conversion, write 1 to MCR[STRIGn].
11. Wait until NIS[CHF*n*] = 1. This indicates that the FIFOs are nearly full.
12. Process the audio data. This can include:
    - Adjusting the CIC gain. For example, divide the 24-bit result by OSR^{ORD} (in this case, C_AC78h) to get a fixed-point decimal value between –1.0 and 1.0. Then, account for the ADC reference voltage to get the real value.

- Using your software to perform additional compensation, low-pass filtering (sharpening), and decimation by 4. See Frequency response after LPF and compensation for results.
- Using your software to perform HPF gain compensation. Frequency response after HPF shows that you must compensate by adding 0.1367 dB.
- Using your software or another module (such as a sample-rate converter) to perform asynchronous data-rate conversion. In this use case, the final data rate is 44.22 kHz, but the audio system requires 44.000 kHz.
- Optionally performing delay compensation or beamforming. You can use a Lagrange polynomial for this.

13. Go to step 11 for the next loop.

47.6.6 Use case: Power meter with Continuous mode

47.6.6.1 Overview

You can use SINC for a power meter.

A typical low-cost power meter can achieve a wide dynamic range (approximately 6 mA to 60 A) using:

- A successive approximation register ADC (SAR ADC), with 12- to 16-bit resolution, to measure typical voltages (for example, 220 V or 380 V) that have a stable wave.
- A sigma-delta ADC (SD ADC), with 20- to 24-bit resolution, to measure current.

For a single-phase power meter, use one SINC channel to measure current and one SAR ADC to measure voltage.

For a three-phase power meter, use three or four SINC channels to measure current and three SAR ADCs to measure voltage. If your SAR ADC supports multiplexed channels, you can use one SAR ADC to sample three voltages at different times.

Ideally, voltage and current measurements must occur at the same time. However, in this use case, simultaneous measurement is hard to achieve. You can use SINC's FILT_CC output to synchronize the measurement time between SINC and SAR ADC.

Schematic shows the power-meter schematic.

47.6.6.2 Schematic

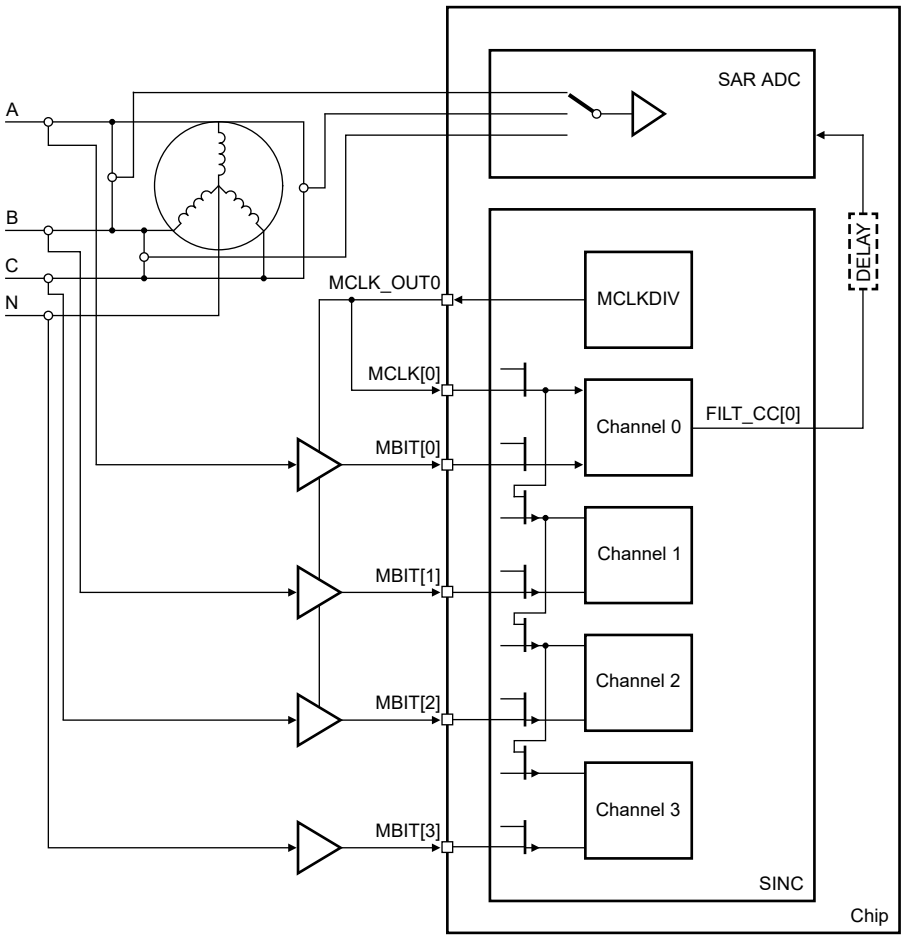


Figure 250. Schematic

47.6.6.3 Considerations

SAR ADC must support switching and must sample three voltage inputs when it receives a trigger.

This use case includes the following delays:

- From the SINC sample time to FILT_CC
- From FILT_CC to the delay block
- From the delay block to the start of SAR ADC's conversion of A in Schematic
- From the end of SAR ADC's conversion of A to the start of conversion B

You can calculate the delay, and you must compensate for that delay.

You can also manage delays by configuring the delay value to be near the SINC sample rate (0.5 × OSR), and ignore SINC's first output. In this case, when SINC completes its second conversion, SAR ADC has also just finished its first sample. Therefore, you can reduce the delay-compensation effort and improve precision by carefully configuring the delay value.

For the exact delay setting, consider SINC's CIC phase response and align it to SAR ADC's analog sample-and-hold time (not including conversion time). CIC phase response is linear, so you can convert it to a delay time, which is about half of OSR in Continuous mode.

47.7 SINC register descriptions

SINC supports a programming model for:

- Configuring the module.
- Storing the FIFO buffer.
- Enabling clock generation.
- Generating triggers.
- Protecting SINC operation.
- Using DMA.

Before you enable SINC or its channels, you must configure certain fields as described in Configure clocking-related fields before enabling SINC and Configure fields before enabling channels.

In the memory map, the following addresses are reserved and do not generate a transfer error if you access them:

- 28h
- 2Ch
- 30h
- 34h

47.7.1 SINC memory map

SINC0 base address: 4010_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0200_007Dh
4h	Parameters (PARAMETER)	32	RW	0070_0508h
8h	Main Control (MCR)	32	RW	0000_0000h
Ch	Normal Interrupt Enable (NIE)	32	RW	0000_0000h
10h	Error Interrupt Enable (EIE)	32	RW	0000_0000h
14h	FIFO And CAD Error Interrupt Enable (FIFOIE)	32	RW	0000_0000h
18h	Normal Interrupt Status (NIS)	32	RW	0000_0000h
1Ch	Error Interrupt Status (EIS)	32	RW	0000_0000h
20h	FIFO And CAD Error Interrupt Status (FIFOIS)	32	RW	0000_0000h
24h	Status (SR)	32	R	001F_0000h
38h	Channel 0 Control (C0CR)	32	RW	0000_0000h
3Ch	Channel 0 Data Rate (C0DR)	32	RW	0000_1800h
40h	Channel 0 Configuration (C0CFR)	32	RW	0020_0000h
44h	Channel 0 Protection (C0PROT)	32	RW	0000_0000h
48h	Channel 0 Bias (C0BIAS)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
4Ch	Channel 0 Low Limit (C0LOLMT)	32	RW	0000_0000h
50h	Channel 0 High Limit (C0HILMT)	32	RW	0000_0000h
54h	Channel 0 Result Data (C0RDATA)	32	R	0000_0000h
58h	Channel 0 Multipurpose Data (C0MPDATA)	32	RW	0000_0000h
5Ch	Channel 0 Advanced Configuration (C0ACFR)	32	RW	0000_0000h
60h	Channel 0 Status (C0SR)	32	RW	0000_0000h
64h	Channel 0 Debug (C0DBGR)	32	R	0000_0000h
68h	Channel 1 Control (C1CR)	32	RW	0000_0000h
6Ch	Channel 1 Data Rate (C1DR)	32	RW	0000_1800h
70h	Channel 1 Configuration (C1CFR)	32	RW	0020_0000h
74h	Channel 1 Protection (C1PROT)	32	RW	0000_0000h
78h	Channel 1 Bias (C1BIAS)	32	RW	0000_0000h
7Ch	Channel 1 Low Limit (C1LOLMT)	32	RW	0000_0000h
80h	Channel 1 High Limit (C1HILMT)	32	RW	0000_0000h
84h	Channel 1 Result Data (C1RDATA)	32	R	0000_0000h
88h	Channel 1 Multipurpose Data (C1MPDATA)	32	RW	0000_0000h
8Ch	Channel 1 Advanced Configuration (C1ACFR)	32	RW	0000_0000h
90h	Channel 1 Status (C1SR)	32	RW	0000_0000h
94h	Channel 1 Debug (C1DBGR)	32	R	0000_0000h
98h	Channel 2 Control (C2CR)	32	RW	0000_0000h
9Ch	Channel 2 Data Rate (C2DR)	32	RW	0000_1800h
A0h	Channel 2 Configuration (C2CFR)	32	RW	0020_0000h
A4h	Channel 2 Protection (C2PROT)	32	RW	0000_0000h
A8h	Channel 2 Bias (C2BIAS)	32	RW	0000_0000h
ACh	Channel 2 Low Limit (C2LOLMT)	32	RW	0000_0000h
B0h	Channel 2 High Limit (C2HILMT)	32	RW	0000_0000h
B4h	Channel 2 Result Data (C2RDATA)	32	R	0000_0000h
B8h	Channel 2 Multipurpose Data (C2MPDATA)	32	RW	0000_0000h
BCh	Channel 2 Advanced Configuration (C2ACFR)	32	RW	0000_0000h
C0h	Channel 2 Status (C2SR)	32	RW	0000_0000h
C4h	Channel 2 Debug (C2DBGR)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
C8h	Channel 3 Control (C3CR)	32	RW	0000_0000h
CCh	Channel 3 Data Rate (C3DR)	32	RW	0000_1800h
D0h	Channel 3 Configuration (C3CFR)	32	RW	0020_0000h
D4h	Channel 3 Protection (C3PROT)	32	RW	0000_0000h
D8h	Channel 3 Bias (C3BIAS)	32	RW	0000_0000h
DCh	Channel 3 Low Limit (C3LOLMT)	32	RW	0000_0000h
E0h	Channel 3 High Limit (C3HILMT)	32	RW	0000_0000h
E4h	Channel 3 Result Data (C3RDATA)	32	R	0000_0000h
E8h	Channel 3 Multipurpose Data (C3MPDATA)	32	RW	0000_0000h
ECh	Channel 3 Advanced Configuration (C3ACFR)	32	RW	0000_0000h
F0h	Channel 3 Status (C3SR)	32	RW	0000_0000h
F4h	Channel 3 Debug (C3DBGR)	32	R	0000_0000h
F8h	Channel 4 Control (C4CR)	32	RW	0000_0000h
FCh	Channel 4 Data Rate (C4DR)	32	RW	0000_1800h
100h	Channel 4 Configuration (C4CFR)	32	RW	0020_0000h
104h	Channel 4 Protection (C4PROT)	32	RW	0000_0000h
108h	Channel 4 Bias (C4BIAS)	32	RW	0000_0000h
10Ch	Channel 4 Low Limit (C4LOLMT)	32	RW	0000_0000h
110h	Channel 4 High Limit (C4HILMT)	32	RW	0000_0000h
114h	Channel 4 Result Data (C4RDATA)	32	R	0000_0000h
118h	Channel 4 Multipurpose Data (C4MPDATA)	32	RW	0000_0000h
11Ch	Channel 4 Advanced Configuration (C4ACFR)	32	RW	0000_0000h
120h	Channel 4 Status (C4SR)	32	RW	0000_0000h
124h	Channel 4 Debug (C4DBGR)	32	R	0000_0000h

47.7.2 Version ID (VERID)

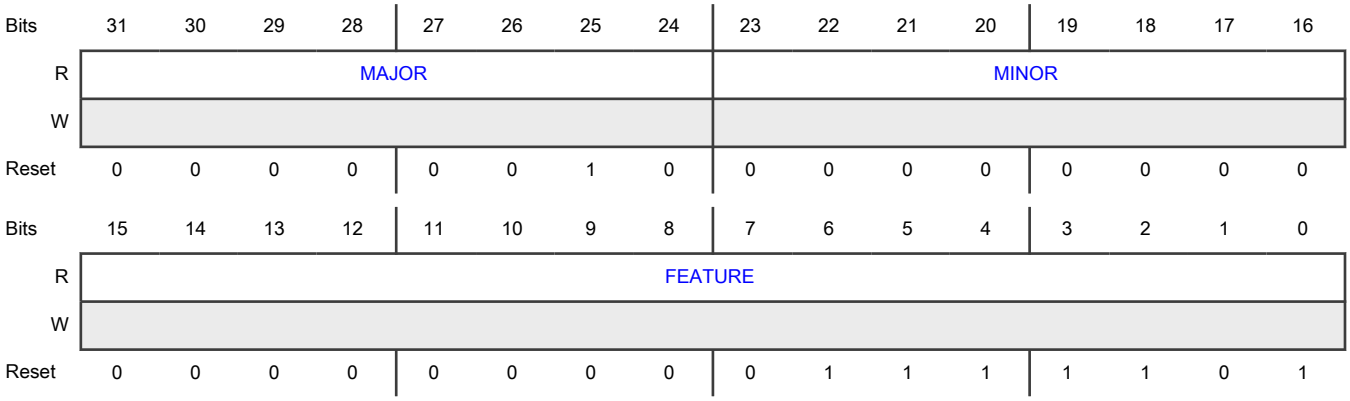
Offset

Register	Offset
VERID	0h

Function

Records the specific SINC version in the chip.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Indicates the major version number of the SINC design. 0000_0001b - 1.x 0000_0010b - 2.x All other values are reserved.
23-16 MINOR	Minor Version Number Indicates the minor version number of the SINC design. 0000_0000b - x.0 All other values are reserved.
15-0 FEATURE	Feature Specification Code Contains a coded value in which each bit represents certain features available in this SINC instance.

Bit position	Feature
0	Feature set
1	Alternate clock selection and divider
2	Asynchronous function clock
4	Reserved
5	Availability of HPF
6	Availability of bias

Field	Function

Bit position	Feature
15–7	Reserved

47.7.3 Parameters (PARAMETER)

Offset

Register	Offset
PARAMETER	4h

Function

Specifies and indicates some of SINC's operating parameters.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				Reserved				0	Reserved		PF_ORD_SEL		0		
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		0	0	FLT_NUM				0				FIFO_DEPTH			
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 —	Reserved
23 —	Reserved
22-21	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
20-19 PF_ORD_SEL	PF Order Select Indicates the maximum order of the PF. 10b - 3 11b - 2 All other values are reserved.
18-16 —	Reserved
15-14 —	Reserved
13 —	Reserved
12 —	Reserved
11-8 FLT_NUM	Filter Channel Number Each filter channel has dedicate input and CIC filters.
7-5 —	Reserved
4-0 FIFO_DEPTH	FIFO Depth Entry numbers for each channel's FIFO.

47.7.4 Main Control (MCR)

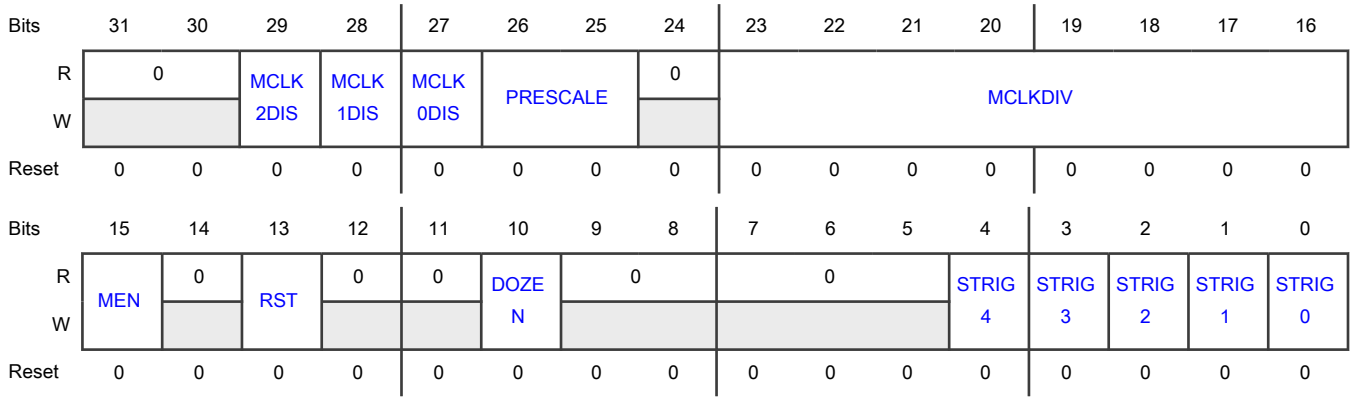
Offset

Register	Offset
MCR	8h

Function

Controls the global features of all channels.

Diagram



Fields

Field	Function
31-30 —	Reserved
29 MCLK2DIS	Disable Modulator Clock 2 Output Works with MEN to disable the MCLK2 output. 0b - Enabled when MEN = 1 1b - Disabled regardless of MEN value
28 MCLK1DIS	Disable Modulator Clock 1 Output Works with MEN to disable the MCLK1 output. 0b - Enabled when MEN = 1 1b - Disabled regardless of MEN value
27 MCLK0DIS	Disable Modulator Clock 0 Output Works with MEN to disable the MCLK0 output. 0b - Enabled when MEN = 1 1b - Disabled regardless of MEN value
26-25 PRESCALE	Prescale Before Clock Divider Specifies the clock divider ratio for the modulator clock. 00b - No prescale 01b - 2 10b - 4 11b - 8
24 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-16 MCLKDIV	Modulator Clock Divider Contributes to the modulator-clock divider ratio according to the following expression: Clock divider ratio = MCLKDIV + 1 The minimum clock divider ratio is 2, and therefore the minimum valid value of MCLKDIV is 1. MCLKDIV = 0 is prohibited. To obtain a 50% duty cycle in the MCLK output, write an odd value to MCLKDIV. 0000_0000b - Prohibited All other values - Added to 1 to specify the clock divider
15 MEN	Master Enable Simultaneously enables all function blocks enabled in their respective registers. Before you write 1 to this field, you must configure other fields as described in Configure clocking-related fields before enabling SINC. 0b - Disables 1b - Enables
14 —	Reserved
13 RST	Software Reset Triggers a reset for functional blocks and some registers as described in Reset. 0b - Do not reset 1b - Reset
12 —	Reserved
11 —	Reserved
10 DOZEN	Doze Or Stop Enable Disables SINC when the chip enters Doze or Stop mode. 0b - Enables 1b - Disables
9-8 —	Reserved
7-5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
4-0 STRIGn	Software Trigger For Channel n 0b - No effect 1b - Trigger

47.7.5 Normal Interrupt Enable (NIE)

Offset

Register	Offset
NIE	Ch

Function

Enables normal interrupt requests.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0			ZCDIE4	ZCDIE3	ZCDIE2	ZCDIE1	ZCDIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			CHFIE4	CHFIE3	CHFIE2	CHFIE1	CHFIE0	0			COCIE4	COCIE3	COCIE2	COCIE1	COCIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-21 —	Reserved
20-16 ZCDIEn	Zero Cross Detected Interrupt Enable Enables the Zero Cross Detected interrupt for filter channel *n*.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disables 1b - Enables
15-13 —	Reserved
12-8 CHFIEn	Data Output Ready Interrupt Enable Enables the Data Output Ready interrupt for PF channel *n*. 0b - Disables 1b - Enables
7-5 —	Reserved
4-0 COCIEn	Conversion Complete Interrupt Enable Enables the Conversion Complete interrupt for PF channel *n*. 0b - Disables 1b - Enables

47.7.6 Error Interrupt Enable (EIE)

Offset

Register	Offset
EIE	10h

Function
Enables error-related interrupt requests.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			HLMTIE4	HLMTIE3	HLMTIE2	HLMTIE1	HLMTIE0	0			LLMTIE4	LLMTIE3	LLMTIE2	LLMTIE1	LLMTIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WLMTIE4	WLMTIE3	WLMTIE2	WLMTIE1	WLMTIE0	0			SCDIE4	SCDIE3	SCDIE2	SCDIE1	SCDIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28-24 HLMTIEn	High Limit Interrupt Enable Enables the High Limit interrupt for filter channel *n*. 0b - Disables 1b - Enables
23-21 —	Reserved
20-16 LLMTIEn	Low Limit Interrupt Enable Enables the Low Limit interrupt for filter channel *n*. 0b - Disables 1b - Enables
15-13 —	Reserved
12-8 WLMTIEn	Window Limit Interrupt Enable Enables the Window Limit interrupt for filter channel *n*. 0b - Disables 1b - Enables
7-5 —	Reserved
4-0 SCDIEn	Short Circuit Detected Interrupt Enable Enables the Short Circuit Detected interrupt for filter channel *n*. 0b - Disables 1b - Enables

47.7.7 FIFO And CAD Error Interrupt Enable (FIFOIE)

Offset

Register	Offset
FIFOIE	14h

Function
Enables FIFO- and CAD-related interrupt requests.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			SATIE4	SATIE3	SATIE2	SATIE1	SATIE0	0			CADIE4	CADIE3	CADIE2	CADIE1	CADIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			FOVFIE4	FOVFIE3	FOVFIE2	FOVFIE1	FOVFIE0	0			FUNFIE4	FUNFIE3	FUNFIE2	FUNFIE1	FUNFIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28-24 SATIEn	Saturation Interrupt Enable Enables the Saturation interrupt for PF channel *n*. 0b - Disables 1b - Enables
23-21 —	Reserved
20-16 CADIEn	Clock Absence Interrupt Enable Enables the Clock Absence interrupt for filter channel *n*. 0b - Disables 1b - Enables
15-13 —	Reserved
12-8 FOVFIEn	FIFO Overflow Interrupt Enable Enables the FIFO Overflow interrupt for PF channel *n*. 0b - Disables 1b - Enables
7-5 —	Reserved
4-0	FIFO Underflow Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
FUNFIEn	Enables the FIFO Underflow interrupt for PF channel *n*. 0b - Disables 1b - Enables

47.7.8 Normal Interrupt Status (NIS)

Offset

Register	Offset
NIS	18h

Function

Contains flags for normal interrupts on each SINC channel.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0			ZCD4	ZCD3	ZCD2	ZCD1	ZCD0
W												W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			CHF4	CHF3	CHF2	CHF1	CHF0	0			COC4	COC3	COC2	COC1	COC0
W				W1C	W1C	W1C	W1C	W1C				W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-21 —	Reserved
20-16 ZCDn	Zero Cross Detected Flag Indicates whether the resulting data on filter channel *n* crossed zero and therefore changed sign. Use C*n*CFR[ZCOP] to configure the type of zero crossing that sets the Zero Cross Detected flag.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
15-13 —	Reserved
12-8 CHFn	Data Output Ready Flag Indicates whether the FIFO from PF channel *n* has overflowed its watermark level and the data is available in Channel n Result Data (C0RDATA - C4RDATA). This flag contributes to generating an interrupt or DMA request if you enabled these by using NIE[CHFIE*n*] or C*n*CR[DMAEN], respectively. SINC sets this flag according to the watermark level. If you enabled DMA requests, SINC clears the flag after the DMA controller finishes the DMA transaction. In other cases—for example, if you enabled interrupts—you clear the flag by writing 1 to it. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No overflow; data not available 1b - Overflow; data available When writing 0b - No effect 1b - Clear the flag
7-5 —	Reserved
4-0 COCn	Conversion Complete Flag Indicates whether one conversion has finished and the data is available in Channel n Result Data (C0RDATA - C4RDATA). **NOTE** This field behaves differently for register reads and writes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When reading 0b - Not finished; data not available 1b - Finished; data available When writing 0b - No effect 1b - Clear the flag

47.7.9 Error Interrupt Status (EIS)

Offset

Register	Offset
EIS	1Ch

Function

Contains flags for error-related interrupts on each SINC channel.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			HLMT4	HLMT3	HLMT2	HLMT1	HLMT0	0			LLMT4	LLMT3	LLMT2	LLMT1	LLMT0
W				W1C	W1C	W1C	W1C	W1C				W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WLMT4	WLMT3	WLMT2	WLMT1	WLMT0	0			SCD4	SCD3	SCD2	SCD1	SCD0
W				W1C	W1C	W1C	W1C	W1C				W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28-24 HLMTn	High Limit Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates whether the conversion result for filter channel *n* exceeded the threshold value specified in C*n*HILMT[HILMT]. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not exceeded 1b - Exceeded When writing 0b - No effect 1b - Clear the flag
23-21 —	Reserved
20-16 LLMTn	Low Limit Flag Indicates whether the conversion result for filter channel *n* exceeded the threshold value specified in C*n*LOLMT[LOLMT]. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not exceeded 1b - Exceeded When writing 0b - No effect 1b - Clear the flag
15-13 —	Reserved
12-8 WLMTn	Window Limit Flag Indicates whether filter channel *n* exceeded its window limit. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not exceeded 1b - Exceeded

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When writing 0b - No effect 1b - Clear the flag
7-5 —	Reserved
4-0 SCDn	Short Circuit Detected Flag Indicates whether SINC detected a short circuit on filter channel *n*. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag

47.7.10 FIFO And CAD Error Interrupt Status (FIFOIS)

Offset

Register	Offset
FIFOIS	20h

Function

Contains flags for interrupts pertaining to FIFO and CAD errors on each SINC channel.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			SAT4	SAT3	SAT2	SAT1	SAT0	0			CAD4	CAD3	CAD2	CAD1	CAD0
W				W1C	W1C	W1C	W1C	W1C				W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			FOVF4	FOVF3	FOVF2	FOVF1	FOVF0	0			FUNF4	FUNF3	FUNF2	FUNF1	FUNF0
W				W1C	W1C	W1C	W1C	W1C				W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28-24 SATn	Saturation Flag Indicates whether PF channel *n* is saturated. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not saturated 1b - Saturated When writing 0b - No effect 1b - Clear the flag
23-21 —	Reserved
20-16 CADn	Clock Absence Flag Indicates whether SINC detected the absence of a clock on filter channel *n*. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Clock present 1b - Clock absent When writing 0b - No effect 1b - Clear the flag
15-13 —	Reserved
12-8 FOVFn	FIFO Overflow Flag Indicates whether a FIFO overflow occurred on PF channel *n*. **NOTE** This field behaves differently for register reads and writes. When reading

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Did not occur 1b - Occurred When writing 0b - No effect 1b - Clear the flag
7-5 —	Reserved
4-0 FUNFn	FIFO Underflow Flag Indicates whether a FIFO underflow occurred on PF channel *n*. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not occur 1b - Occurred When writing 0b - No effect 1b - Clear the flag

47.7.11 Status (SR)

Offset

Register	Offset
SR	24h

Function

Presents the status of each SINC channel and modulator clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				Reserved	MCLKRDY2	MCLKRDY1	MCLKRDY0	0			FIFOEMP...	FIFOEMP...	FIFOEMP...	FIFOEMP...	FIFOEMP...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			CHRDY4	CHRDY3	CHRDY2	CHRDY1	CHRDY0	0			CIP4	CIP3	CIP2	CIP1	CIP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27 —	Reserved
26 MCLKRDY2	Modulator Clock 2 Ready Indicates the readiness of modulator clock 2. SINC changes this field to 1 when both of the following conditions are true: • MCR[MEN] = 1. • Seven output clock cycles have passed. 0b - Not ready 1b - Ready
25 MCLKRDY1	Modulator Clock 1 Ready Indicates the readiness of modulator clock 1. SINC changes this field to 1 when both of the following conditions are true: • MCR[MEN] = 1. • Seven output clock cycles have passed. 0b - Not ready 1b - Ready
24 MCLKRDY0	Modulator Clock 0 Ready Indicates the readiness of modulator clock 0. SINC changes this field to 1 when both of the following conditions are true:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• MCR[MEN] = 1. • Seven output clock cycles have passed. 0b - Not ready 1b - Ready
23-21 —	Reserved
20-16 FIFOEMPTYn	FIFO Empty Indicates whether the FIFO for filter channel *n* is empty. 0b - Not empty 1b - Empty
15-13 —	Reserved
12-8 CHRDYn	Channel Ready For Conversion Indicates whether filter channel *n* is ready for conversion. SINC changes this field to 1 when all of these conditions are true: • You have enabled a filter. • MCR[MEN] = 1. • C*n*CR[CHEN] = 1. • The FSM of the associated filter accepts the enable signals. • If you are using Manchester decoding, the valid clock edges are synchronized. **NOTE** If you write 0 to C*n*CR[PFEN] to break the ongoing conversion when selecting parallel or serial data, SINC preserves the value of the corresponding CHRDY*n* field for two more cycles. 0b - Not ready 1b - Ready
7-5 —	Reserved
4-0 CIPn	Conversion In Progress Indicates whether a conversion is in progress on PF channel *n*. 0b - Not in progress 1b - In progress

47.7.12 Channel n Control (C0CR - C4CR)

Offset

Register	Offset
C0CR	38h
C1CR	68h
C2CR	98h
C3CR	C8h
C4CR	F8h

Function

Configures various channel features.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0		0		DBGSEL				0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FIFOEN	LMTEN	ZCDEN	0		CADEN	SCDEN	0				DMAEN	0	PFEN	CHEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28 —	Reserved
27-26	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
25-24 —	Reserved
23-20 DBGSEL	Debug Output Selection Selects the instantaneous data that you want to read from Channel n Debug (C0DBGR - C4DBGR). 0000b - Final data from the PF (24 bits) 0001b - Offset data (24 bits) 0010b - Shifted data from the PF (24 bits) 0011b - DC remover (HPF) data (32 bits) 0100b - Raw data from the PF's CIC filter 0110b - Historical data from SCD 0111b - Data from the Manchester decoder 1000b - Data from CAD 1001b - Number of available entries in the FIFO 1010b - Status of the parallel or serial data converter All other values are reserved.
19-16 —	Reserved
15 —	Reserved
14 FIFOEN	FIFO Enable Enables FIFO transfers for the PF. 0b - Disables 1b - Enables
13 LMTEN	Limit Enable Enables a limit check for the PF. 0b - Disables 1b - Enables
12 ZCDEN	Zero Cross Detect Enable Enables zero-cross detection for the PF. 0b - Disables

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enables
11-10 —	Reserved
9 CADEN	Clock Absence Detect Enable Enables the channel's clock-absence detection. 0b - Disables 1b - Enables
8 SCDEN	Short Circuit Detect Enable Enables the channel's short-circuit detection. 0b - Disables 1b - Enables
7-4 —	Reserved
3 DMAEN	DMA Enable Enables primary DMA transfers when the channel's FIFO exceeds its watermark (C*n*CFR[FIFOWMK]). 0b - Disables 1b - Enables
2 —	Reserved
1 PFEN	PF Enable Enables the channel's PF. 0b - Disables 1b - Enables
0 CHEN	Channel Enable Enables the channel and its subfunctions (PF and SCD). SINC enables the channel first and the channel's subfunctions second. Therefore, before you write 1 to CHEN, you must configure other fields as described in Configure fields before enabling channels. 0b - Disables 1b - Enables

47.7.13 Channel n Data Rate (C0DR - C4DR)

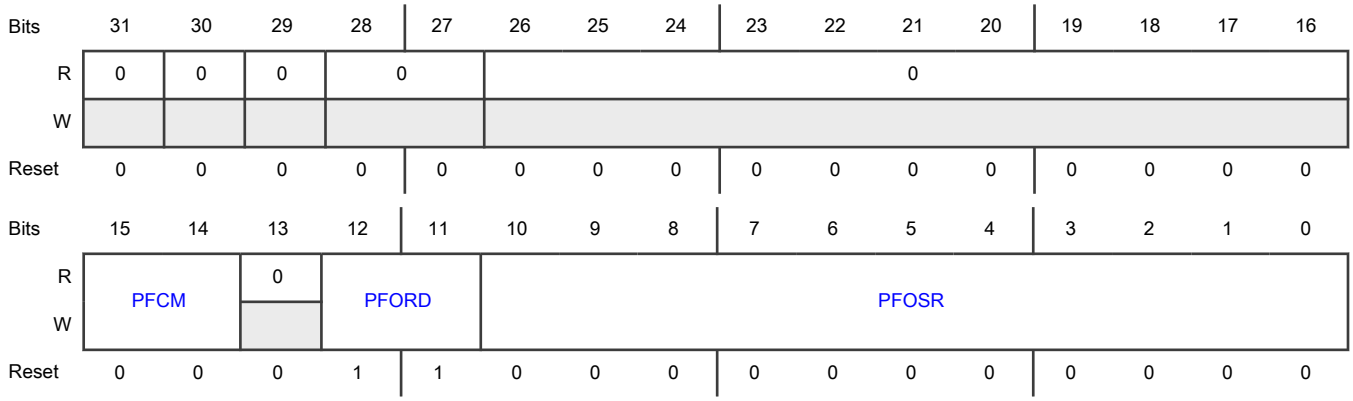
Offset

Register	Offset
C0DR	3Ch
C1DR	6Ch
C2DR	9Ch
C3DR	CCh
C4DR	FCh

Function

Controls functions related to data rate.

Diagram



Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28-27 —	Reserved
26-16	Reserved

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>—</td><td></td></tr>
<tr><td>15-14
PFCM</td><td>PF Conversion Mode
Specifies the conversion mode as described in the following table.
<table><tr><th>Mode</th><th>Description</th></tr><tr><td>Single</td><td>One conversion that follows an edge- or level-triggering event.</td></tr><tr><td>Continuous</td><td>Multiple conversions that follow a triggering event. A new triggering event cancels and restarts conversion.</td></tr><tr><td>Always</td><td>Multiple conversions that follow the first triggering event. SINC ignores the next triggering event.</td></tr><tr><td>Fixed number</td><td>Fixed number conversions that follow the first triggering event. A new triggering event cancels and restarts conversion.</td></tr></table>
00b - Single
01b - Continuous
10b - Always
11b - Fixed number</td></tr>
<tr><td>13
—</td><td>Reserved</td></tr>
<tr><td>12-11
PFORD</td><td>PF Order
Defines the primary CIC filter order.
00b - FastSinc
01b - First order
10b - Second order
11b - Third order</td></tr>
<tr><td>10-0
PFOSR</td><td>PF OSR
Controls the PF OSR as described in Equation for OSR.
The minimum permissible value is 3. Lower values produce unpredictable results.
The maximum permissible value depends on PFORD and the desired data format (CnCFR[RDFMT]), as shown in the following table. Higher values produce incorrect results. It is essential to provide proper shifter value to ensure final results are within 24 bits.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function

PFORD	Data format	Maximum PFOSR	Minimum right shift bits at maximum PFOSR case
3d	Signed	1289	8
	Unsigned	1624	8
Any other value	Signed or unsigned	2047	0

47.7.14 Channel n Configuration (C0CFR - C4CFR)

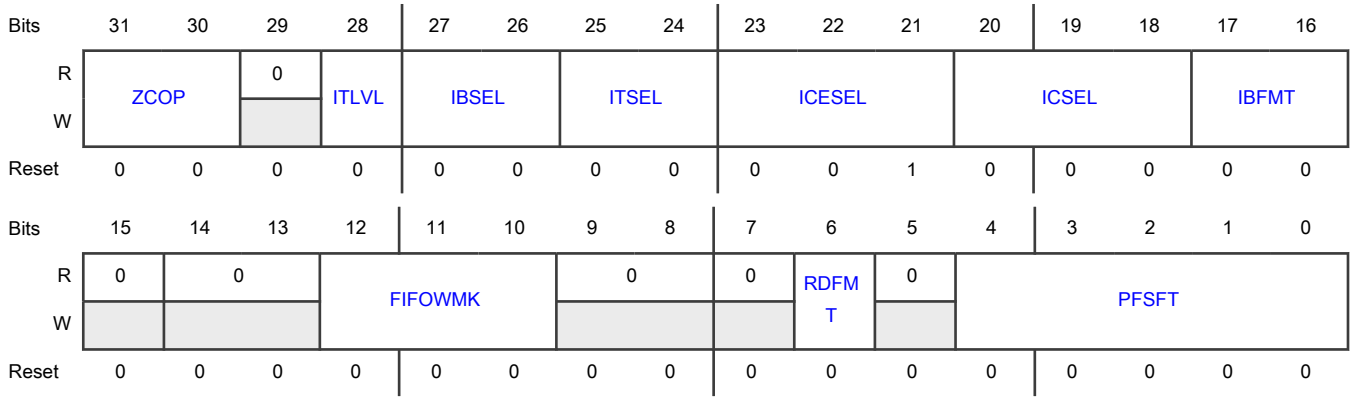
Offset

Register	Offset
C0CFR	40h
C1CFR	70h
C2CFR	A0h
C3CFR	D0h
C4CFR	100h

Function

Configures additional channel features.

Diagram



Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31-30
ZCOP</td><td>Zero Cross Option
Specifies the type of zero crossing that SINC detects.
00b - Both rise and fall
01b - Fall
10b - Rise
All other values are reserved.</td></tr>
<tr><td>29
—</td><td>Reserved</td></tr>
<tr><td>28
ITLVL</td><td>Input Trigger Level Type
Specifies the input trigger level type as described in the following table.
<table><tr><th>ITLVL</th><th>What cancels the current conversion</th><th>What happens in Single or Continuous mode</th></tr><tr><td>0</td><td>Next edge event</td><td>Restart</td></tr><tr><td>1</td><td>Deassertion</td><td>Stop filter function</td></tr></table>
0b - Edge
1b - Level</td></tr>
<tr><td>27-26
IBSEL</td><td>Input Bit Select
Defines the modulate bit source selection.
00b - External bitstream from the MBIT[n] signal
01b - Alternate internal bitstream from the INP[n] signal
11b - Grouped bitstream shared with an adjacent channel; the adjacent channel's IBSEL field determines the input
All other values are reserved.</td></tr>
<tr><td>25-24
ITSEL</td><td>Input Trigger Select
Defines the filter trigger source selection.
00b - Software
01b - Hardware trigger dedicated to the channel
11b - Grouped trigger shared with an adjacent channel; the adjacent channel's ITSEL field determines the trigger
All other values are reserved.</td></tr>
<tr><td>23-21
ICESEL</td><td>Input Clock Edge Select</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Defines the modulate clock edge selection. When selected edge occurs, the modulate bit are sampled for further process. 001b - Positive edge 010b - Negative edge 011b - Both edges 100b - Every other odd positive edge 101b - Every other even positive edge 110b - Every other odd negative edge 111b - Every other even negative edge All other values are reserved.
20-18 ICSEL	Input Clock Select Defines the modulate clock source selection. 000b - MCLK_OUT0 with internal routeback 001b - MCLK_OUT1 with internal routeback 010b - MCLK_OUT2 with internal routeback 011b - External modulator clock dedicated to this channel 111b - Grouped clock shared with an adjacent channel; the adjacent channel's ICSEL field determines the input clock All other values are reserved.
17-16 IBFMT	Input Bit Format Defines the input modulate bit format. 00b - External bitstream from the MBIT[n] signal 01b - External Manchester code; ICESEL selects the rise or fall decoder 10b - Internal 16-bit parallel data from MPDATA 11b - Internal 32-bit serial data from MPDATA
15 —	Reserved
14-13 —	Reserved
12-10 FIFOWMK	FIFO Watermark Specifies the FIFO watermark. When the number of entries in the FIFO exceeds this value, SINC sets the Data Output Ready flag (see NIS[CHF*n*]).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If you enable interrupt requests (write 1 to NIE[CHFIE*n*]), SINC also generates an interrupt request. If you enable DMA (write 1 to C*n*CR[DMAEN]), SINC also generates a DMA request.
9-8 —	Reserved
7 —	Reserved
6 RDFMT	Result Data Format Result data are 24bit width, located at CnRDATA[31:8]. This field controls the signed or unsigned format. If signed format, CnRDATA[31] indicates the negative or positive value. 0b - Left justified, signed 1b - Left justified, unsigned
5 —	Reserved
4-0 PFSFT	PF Shift Shifts the PF data for the correct 24-bit precision. See Shift for details. Bits 0–3 specify the number of bits to shift the data. Bit 4 specifies the shift direction: 0b - Right 1b - Left

47.7.15 Channel n Protection (C0PROT - C4PROT)

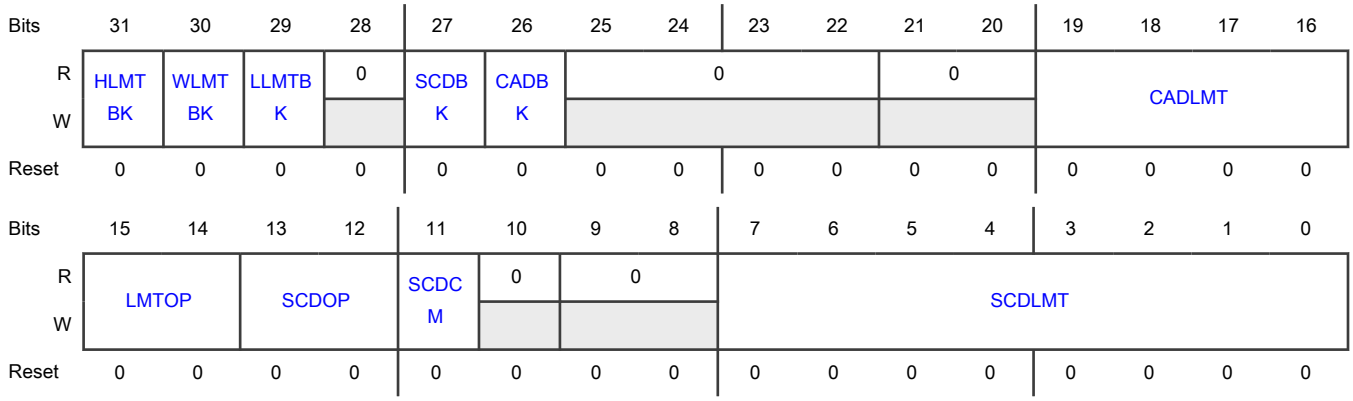
Offset

Register	Offset
C0PROT	44h
C1PROT	74h
C2PROT	A4h
C3PROT	D4h
C4PROT	104h

Function

Configures the SINC functions related to protection (SCD, CAD, and limits).

Diagram



Fields

Field	Function
31 HLMTBK	High Limit Break Signal Enables the automatic assertion of the BREAK_HIGH signal (see External signals) when SINC detects a high-limit event on the associated channel. 0b - Disables 1b - Enables
30 WLMTBK	Window Limit Break Signal Enables the automatic assertion of the BREAK_WIN signal (see External signals) when SINC detects a window-limit event on the associated channel. 0b - Disables 1b - Enables
29 LLMTBK	Low Limit Break Signal Enables the automatic assertion of the BREAK_LOW signal (see External signals) when SINC detects a low-limit event on the associated channel. 0b - Disables 1b - Enables
28 —	Reserved
27 SCDBK	SCD Break Signal Enables the automatic assertion of the BREAK_SCD signal (see External signals) when SINC detects an SCD event on the associated channel. 0b - Disables 1b - Enables
26	CAD Break Signal

Table continues on the next page...

Table continued from the previous page...

Field	Function
CADBK	Enables the automatic assertion of the BREAK_CAD signal (see External signals) when SINC detects a CAD event on the associated channel. 0b - Disables 1b - Enables
25-22 —	Reserved
21-20 —	Reserved
19-16 CADLMT	CAD Limit Threshold Specifies the threshold value for the CAD counter. The CAD counter tracks the number of clock cycles during which SINC does not detect a clock. If that number exceeds CADLMT, a CAD event occurs on the associated channel. You disable CAD by writing 0 to CADLMT. 0000b - Disables CAD All other values - Threshold value
15-14 LMTOP	Limit Detection Option Specifies whether SINC compares the filter sample value to: • The low limit (C*n*LOLMT[LOLMT]) • The high limit (C*n*HILMT[HILMT]) • Both the high and low limits • A window of values between the low and high limits See Limit detection operations for more information. 00b - Both high and low limits 01b - High limit 10b - Low limit 11b - Windowed value
13-12 SCDOP	SCD Option Specifies which repeating bit value increments the SCD counter. 00b - Both 0 and 1 01b - Only 1 10b - Only 0 11b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 SCDCM	SCD Conversion Mode Specifies when SCD operates. 0b - Constantly when CnCR[CHEN] = MCR[MEN] = 1 1b - Only when the PF is performing a conversion
10 —	Reserved
9-8 —	Reserved
7-0 SCDLMT	SCD Limit Threshold Specifies the threshold value for the SCD counter. The SCD counter tracks the number of received bits with the same repeating value (always 0 or always 1). If that number exceeds SCDLMT, an SCD event occurs on the associated channel. 0000_0000b-0000_0001b - Disables SCD All other values - Threshold value

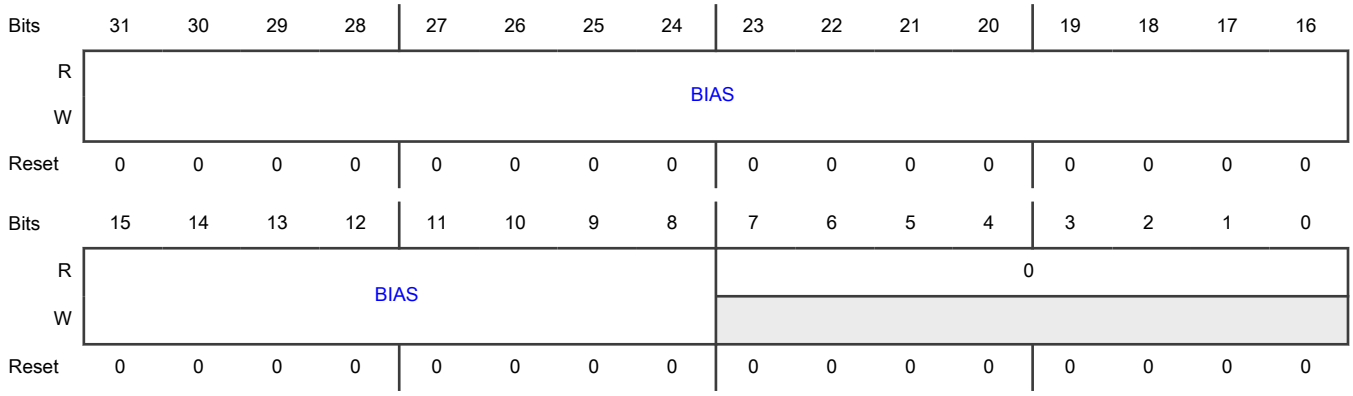
47.7.16 Channel n Bias (C0BIAS - C4BIAS)

Offset

Register	Offset
C0BIAS	48h
C1BIAS	78h
C2BIAS	A8h
C3BIAS	D8h
C4BIAS	108h

Function
Configures the filter bias.

Diagram



Fields

Field	Function
31-8 BIAS	Bias Value Specifies the bias offset for this channel's PF. After the PF shifts its data, the filter subtracts this bias value from the data. Bit 23 of this field (bit 31 within the register) specifies the bias sign: 0b - Positive 1b - Negative
7-0 —	Reserved

47.7.17 Channel n Low Limit (C0LOLMT - C4LOLMT)

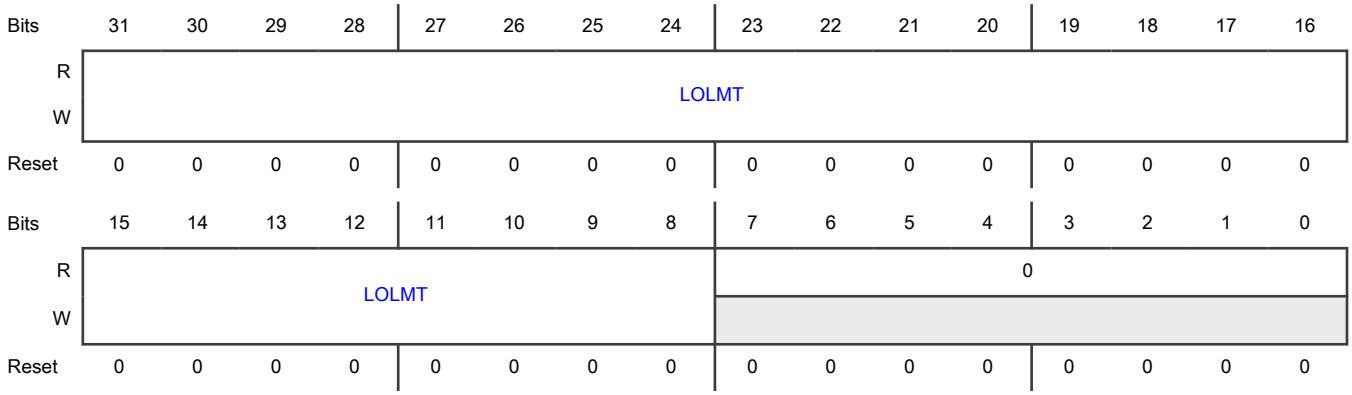
Offset

Register	Offset
C0LOLMT	4Ch
C1LOLMT	7Ch
C2LOLMT	ACh
C3LOLMT	DCh
C4LOLMT	10Ch

Function

Specifies the low-limit threshold value.

Diagram



Fields

Field	Function
31-8 LOLMT	Low Limit Threshold Specifies the low-limit threshold value. When the data exceeds this threshold value, a low-limit event occurs. C*n*CFR[RDFMT] determines the limit threshold format (signed or unsigned). **NOTE** Ensure that LOLMT is lower than C*n*HILMT[HILMT]. Otherwise, the low-limit threshold does not work.
7-0 —	Reserved

47.7.18 Channel n High Limit (C0HILMT - C4HILMT)

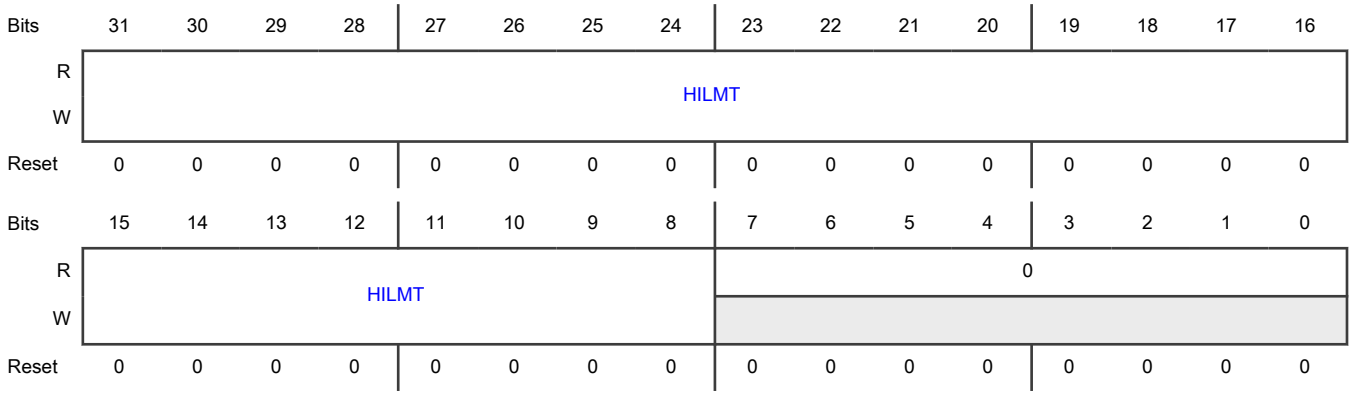
Offset

Register	Offset
C0HILMT	50h
C1HILMT	80h
C2HILMT	B0h
C3HILMT	E0h
C4HILMT	110h

Function

Specifies the high-limit threshold value.

Diagram



Fields

Field	Function
31-8 HILMT	High Limit Threshold Specifies the high-limit threshold value. When the data exceeds this threshold value, a high-limit event occurs. C*n*CFR[RDFMT] determines the limit threshold format (signed or unsigned). **NOTE** Ensure that C*n*LOLMT[LOLMT] is lower than HILMT. Otherwise, the low-limit threshold does not work.
7-0 —	Reserved

47.7.19 Channel n Result Data (C0RDATA - C4RDATA)

Offset

Register	Offset
C0RDATA	54h
C1RDATA	84h
C2RDATA	B4h
C3RDATA	E4h
C4RDATA	114h

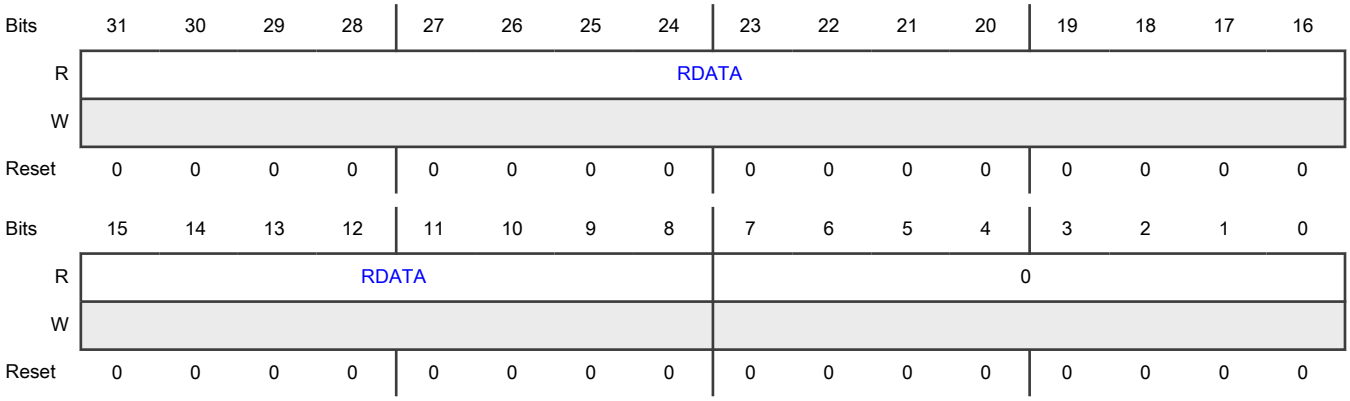
Function

Returns data from the FIFO.

NOTE

NXP recommends that you use a 32-bit access to read this register. If you use a byte access to read this register, you must read the highest byte last, and this increases the FIFO level.

Diagram



Fields

Field	Function
31-8 RDATA	Result Data Contains data from the FIFO. Bit 23 of this field (bit 31 within the register) serves as the sign bit if you specify the signed-data format (C*n*CFR[RDFMT] = 0). When you read this field, be aware of the following circumstances:

Conditions when you read RDATA	Result
SINC has not yet completed the conversion	One of the following occurs: • RDATA contains all zeros. • SINC triggers a FIFO underflow error.
FIFO is disabled (C*n*CR[FIFOEN] = 0)	RDATA contains the last conversion result. You must poll the C*n*SR[SRDS] flag, and if that flag is set after you read RDATA, you must read RDATA again.

7-0 —	Reserved

47.7.20 Channel n Multipurpose Data (C0MPDATA - C4MPDATA)

Offset

Register	Offset
C0MPDATA	58h
C1MPDATA	88h
C2MPDATA	B8h
C3MPDATA	E8h
C4MPDATA	118h

Function

Contains multipurpose data as specified in C*n*CFR[IBFMT].

You can use this register for a parallel bitstream, a serial bitstream, or a Manchester decoder threshold as described in the following table.

Table 425. MPDATA content options

Data type	C*n*CFR[IBFMT]	Description
Manchester	01b	Bits 10–0 contain the Manchester decoder threshold value.
Parallel	10b	Bits 15–0 contain the parallel 16-bit data. You can write to MPDATA only when parallel or serial data is ready (C*n*SR[PSRDY] = 1). Otherwise, a transfer error occurs.
Serial	11b	Bits 31–0 contain the serial data. You can write to MPDATA only when parallel or serial data is ready (C*n*SR[PSRDY] = 1) or MPDATA is not completely full. Otherwise, a transfer error occurs. **NOTE** Allow write MPDATA when C*n*SR[PSRDY] = 0, give you the chance if keep the remain bit for next conversion or overwrite with new data. In this case, the write are only allowed one time. **NOTE** MPDATA are shift out,you can read back for debug. But in corner case, last bit does not shift out while remain bit are zero.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MPDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MPDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MPDATA	Multipurpose Data Contains multipurpose data as described in Table 425. **NOTE** if used as Serial Data, MCLKDIV at least divider by 4. And make sure MCLKOUT[0] period bigger than 2x bus clock period plus 2x func clock period

47.7.21 Channel n Advanced Configuration (C0ACFR - C4ACFR)

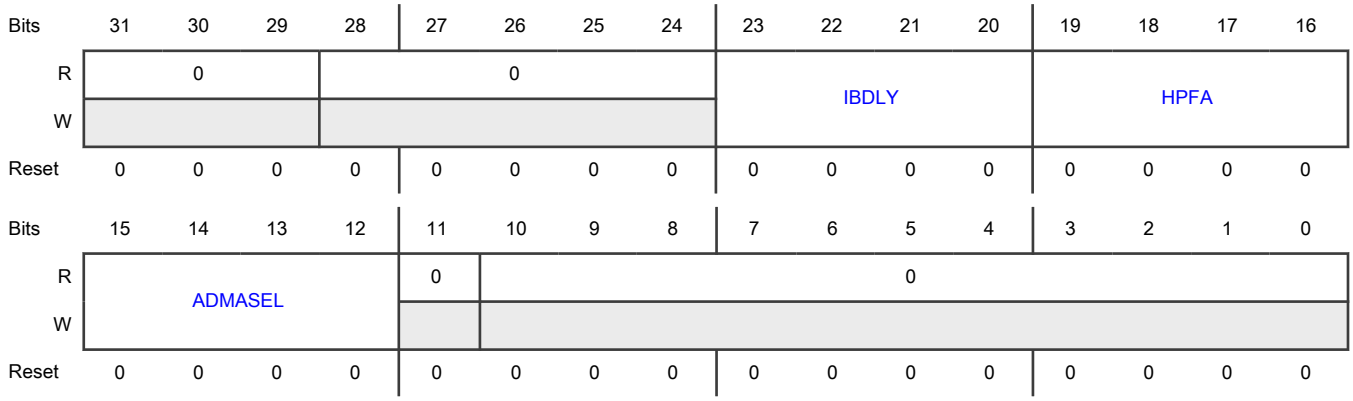
Offset

Register	Offset
C0ACFR	5Ch
C1ACFR	8Ch
C2ACFR	BCh
C3ACFR	ECh
C4ACFR	11Ch

Function

Configures the SINC functional subblocks.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-24 —	Reserved
23-20 IBDLY	Input Modulator Bitstream Delay Specifies the delay, in prescale clock cycles, for sampling modulator bitstreams after an EMCLK edge. A value of 0 disables the delay and SINC uses EMCLK to sample EMBIT. **NOTE** When you choose the delay, ensure that it does not exceed the next EMCLK edge. For example, if the EMCLK width is 13 cycles, the delay value cannot exceed 12d. Values greater than the next EMCLK edge produce unpredictable results. 0000b - Disabled All other values - Delay in clock cycles
19-16 HPFA	HPF DC Remover Alpha Coefficient Specifies the HPF alpha coefficient or disables HPF as described in the following table.

HPFA	Result
0	Disable HPF
1	Coefficient = $1 - 2^{-5}$
2	Coefficient = $1 - 2^{-6}$
3	Coefficient = $1 - 2^{-7}$
4	Coefficient = $1 - 2^{-8}$
...	

Field	Function
	<table><tr><th>HPFA</th><th>Result</th></tr><tr><td>*m*</td><td>Coefficient = $1 - 2^{-4-m}$</td></tr></table>
15-12 ADMASEL	Alternate DMA Source Selection Selects the trigger source for alternate DMA. 0000b - Alternate DMA disabled 0001b - PF conversion complete 0010b - PF data output ready 0011b - Zero crossing detected 0100b - Short circuit detected 0101b - Window limit detected 0110b - Low limit detected 0111b - High limit 1000b - FIFO underflow 1001b - FIFO overflow 1010b - Clock absence 1011b - Saturation
11 —	Reserved
10-0 —	Reserved

47.7.22 Channel n Status (C0SR - C4SR)

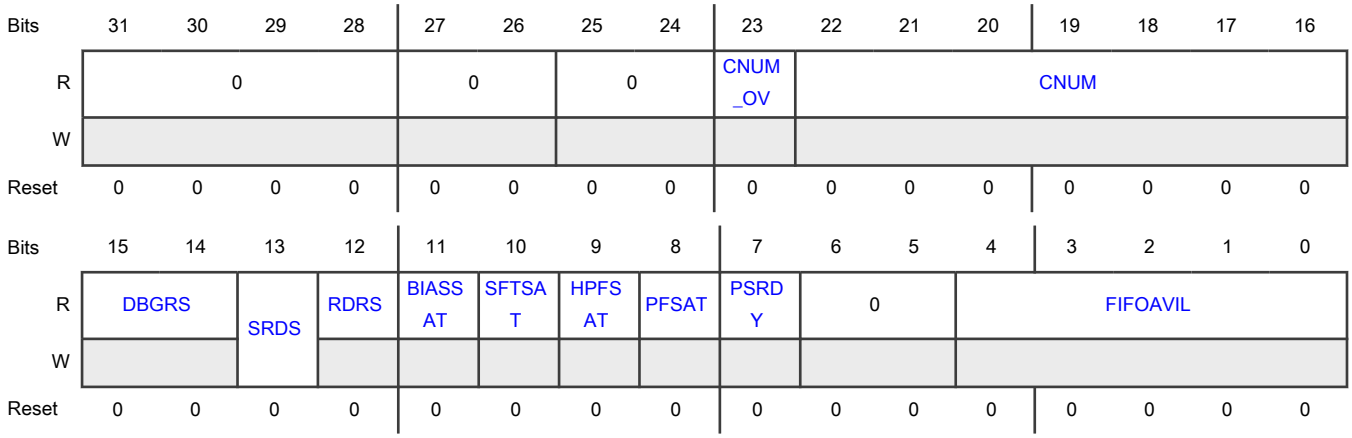
Offset

Register	Offset
C0SR	60h
C1SR	90h
C2SR	C0h
C3SR	F0h
C4SR	120h

Function

Provides channel-status information.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-26 —	Reserved
25-24 —	Reserved
23 CNUM_OV	Overflow In Number Of Conversions Indicates when the number of conversions exceeds 127. When this occurs, SINC changes CNUM_OV to 1. After resynchronization, SINC changes CNUM_OV to 0. 0b - No overflow 1b - Overflow
22-16 CNUM	Number Of Conversions In coordination with CNUM_OV, specifies the number of conversions:

Mode	Event
Continuous	After the start of a trigger
Single	After the enablement of conversion

SINC counts the number of conversions internally and modifies CNUM and CNUM_OV as follows:

When the number of conversions is:	SINC does this:
127 or fewer	Stores the number of conversions in CNUM

Field	Function
	<table><tr><th>When the number of conversions is:</th><th>SINC does this:</th></tr><tr><td>128 or more</td><td>• Stores [(number of conversions) mod 128] in CNUM • Changes CNUM_OV to 1</td></tr></table>
15-14 DBGRS	Debug Data Read Status Indicates whether the debug data is valid. When this value is 0, the CIC, HPF, shift, and offset outputs are all correct. 00b - Valid 01b-11b - Invalid
13 SRDS	Start Read Debug Data Sync Starts the debug data latch procedure and reports the procedure status. When you write 1 to this field, SINC starts the procedure and changes this field to 0. When the field becomes 0 again, you can safely read the debug data (Channel n Debug (C0DBGR - C4DBGR)). **NOTE** This field behaves differently for register reads and writes. When reading 0b - Data valid 1b - Procedure in progress When writing 0b - No effect 1b - Starts the procedure
12 RDRS	Result Data Direct Read Status When you disable the FIFO to read channel data from it, indicates that the data in Channel n Result Data (C0RDATA - C4RDATA) is stable and valid. Before this value is 1, the data may not be stable. Therefore, to obtain stable data, do one of the following: • Wait until the conversion completes, then read C*n*RDATA. • When you read data, ensure that RDRS = 0. 0b - Valid 1b - Invalid
11 BIASSAT	Bias Saturation Flag Indicates whether bias saturation occurred. 0b - Did not occur

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Occurred
10 SFTSAT	Shift Saturation Flag Indicates whether shift saturation occurred. 0b - Did not occur 1b - Occurred
9 HPFSAT	HPF Saturation Flag Indicates whether HPF saturation occurred. 0b - Did not occur 1b - Occurred
8 PFSAT	Primary CIC Saturation Flag Indicates whether primary CIC filter saturation occurred. 0b - Did not occur 1b - Occurred
7 PSRDY	Parallel or Serial Data Ready Indicates whether Channel n Multipurpose Data (C0MPDATA - C4MPDATA) is ready for you to write parallel or serial data. 0b - Not ready 1b - Ready
6-5 —	Reserved
4-0 FIFOAVIL	FIFO Available Data Indicates the number of remaining data entries in the FIFO.

47.7.23 Channel n Debug (C0DBGR - C4DBGR)

Offset

Register	Offset
C0DBGR	64h
C1DBGR	94h
C2DBGR	C4h
C3DBGR	F4h
C4DBGR	124h

Function

Contains the instantaneous debug data and status for debugging each channel's subblocks.

You select the data by writing to C*n*CR[DBGSEL], as shown in Details of debug data. For example:

- If you write 3d to C*n*CR[DBGSEL], SINC offers you 32 bits of data from the DC remover (HPF).
- If you write 4d to C*n*CR[DBGSEL], SINC offers you raw data from the PF's CIC filter.

The instantaneous data is read-only, and the contents are for each subblock.

See Reading asynchronous data safely and Transferring data between domains safely for the procedures to manage data safely.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DBGDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DBGDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DBGDATA	Debug Data Returns the debug data requested in C*n*CR[DBGSEL]. Details of debug data shows the structure of the debug data.

Chapter 48
Standard Counter/Timers (CTIMER)

48.1 Chip-specific CTIMER information

Table 426. Reference links to related information

Topic	Related module	Reference
Full description	CTIMER	CTIMER
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See TIMERnTRIG registers in INPUTMUX

This chip includes 5x standard 32-bit capture/match/pwm timers. Each timer has up to 4x capture and 4x match registers with corresponding inputs and outputs.

NOTE

I2S and USB signals to feed into timers at capture inputs.

48.1.1 Module instances

This device has five instances of the CTIMER module, CTIMER0, CTIMER1, CTIMER2, CTIMER3, CTIMER4.

48.1.2 Signals

The tables below give a summary of each of the Timer/Counter related pins and the recommended PORT settings. Also note that different part number and package variations may provide different CTIMER related pin functions.

Table 427. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAP3:0 CTIMER1_CAP3:0 CTIMER2_CAP3:0 CTIMER3_CAP3:0 CTIMER4_CAP3:0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock. For more details, see CTCR register.
CTIMER0_MAT3:0 CTIMER1_MAT3:0 CTIMER2_MAT3:0 CTIMER3_MAT3:0 CTIMER4_MAT3:0	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel.

48.1.3 Initialization

1. Select a clock source for the CTIMER using the CTIMERCLKSEL register.
2. Enable the clock to the CTIMER via CTIMERCLKSELn[SEL]. This enables the register interface and the peripheral function clock.
3. Clear the CTIMER peripheral reset using the Peripheral Reset Control 0 (PRESETCTRL0) and Peripheral Reset Control 1 (PRESETCTRL1) registers.
4. Each CTIMER provides interrupts to the NVIC. See MCR and CCR registers in the CTIMER register section for match and capture events. For interrupt connections, see NVIC interrupt assignments.
5. Select timer pins and pin modes as needed through the relevant PORT registers.
6. The CTIMER DMA request lines are connected to the DMA trigger inputs via the DMAC0_ITRIG_INMUX registers (See INPUTMUX). Note that timer DMA request outputs are connected to DMA trigger inputs.

48.1.4 External global enable

When the timer is enabled by the CTIMER global start register in SYSCON, (CTIMER Global Start Enable (CTIMERGLOBALSTARTEN)), it is equivalent to writing 1 to the TCR[CEN] bit in CTIMER. The external global start register is used to enable multiple timers at the same time. This operation is allowed by writing a 1 to each Timer's TCR[AGCEN] bit. If the Timer control register's CEN bit is already 1, the external global start enable register has no effect (See SYSCON[CTIMERGLOBALSTARTEN]).

For more information refer CTIMER global start enable register (See SYSCON).

48.1.5 Peripheral input multiplexers

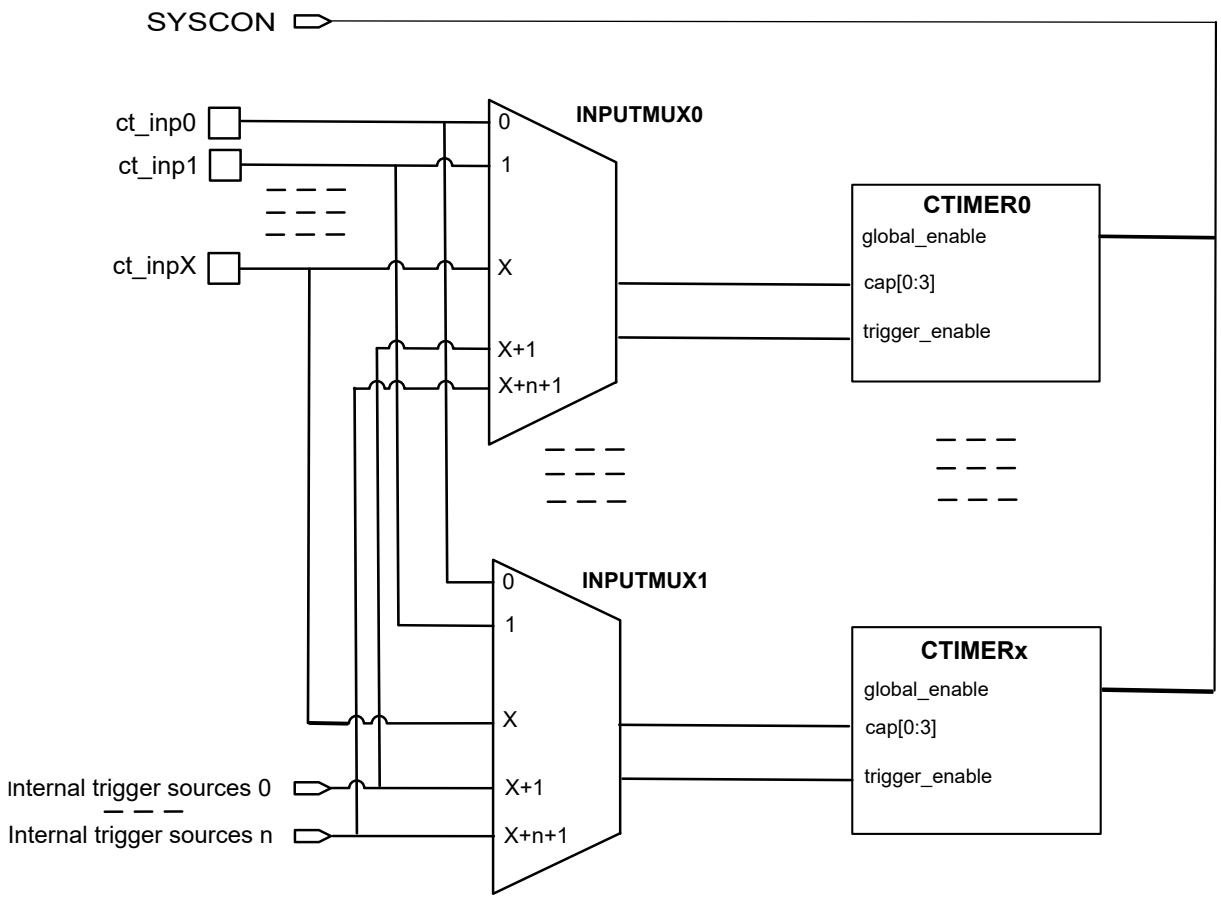


Figure 251. Peripheral input multiplexers for CTIMER

48.2 Overview

Each CTIMER is designed to count cycles of the CTIMER function clock or an externally supplied clock. A CTIMER can optionally generate an interrupt or perform other actions at specified timer values based on the settings of Match (MR0 - MR3). Each CTIMER also includes capture input pins (see Capture mode) to capture the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length. All match registers can optionally be auto-reloaded from a companion shadow register (see Match Shadow (MSR0 - MSR3)) whenever the counter is reset to zero. This feature permits modifying the match values for the next counter cycle without disrupting PWM waveforms during the current cycle. When enabled, match reload occurs whenever the counter is reset, either because of a match event or by writing 1 to TCR[CRST].

48.2.1 Block diagram

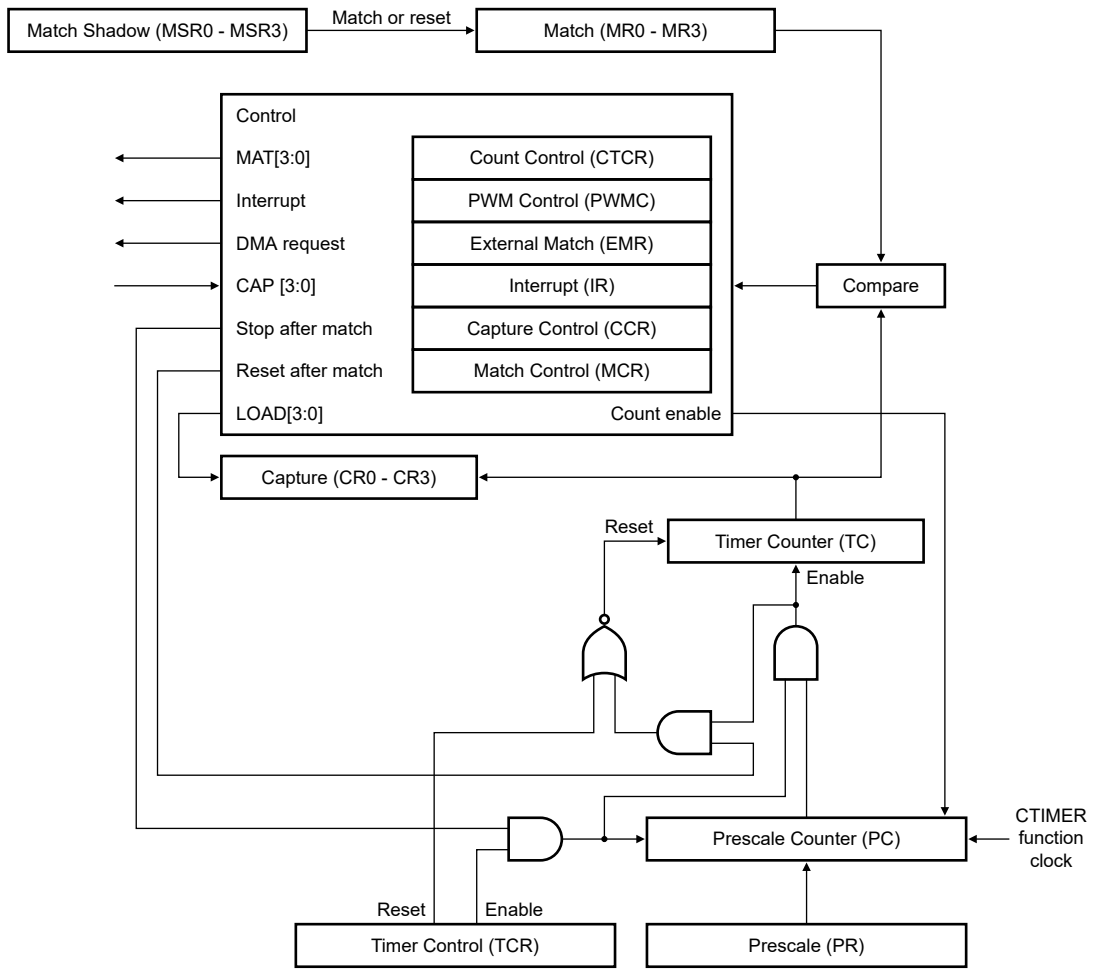


Figure 252. Block diagram

48.2.2 Features

- Contains a 32-bit CTIMER with a programmable 32-bit prescaler (the timers include external capture and match pin connections)
- Contains Timer Control (TCR) and External trigger enable, either of which you can use to enable each timer (you can also use an external global start enable register to globally enable one or more timers)
- Performs counter or timer operations
- Selects the function clocks that may be asynchronous to other system clocks
- Includes up to four 32-bit captures (see Capture mode) to take a snapshot of the timer value when an input signal transitions (a capture event may also optionally generate an interrupt, and the number of capture inputs available for each timer on a chip may vary)
- Enables you to configure the timer and prescaler to be cleared on a designated capture event (this feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse, capturing the timer value on the trailing edge)
- Includes Match (MR0 - MR3) to allow:
  - Continuous operation with optional interrupt generation after a match occurs

  - Optional auto-reload from Match Shadow (MSR0 - MSR3) when the counter is reset
  - Stop timer after a match with optional interrupt generation
  - Reset timer after a match with optional interrupt generation
- Generates up to four external outputs corresponding to Match (MR0 - MR3), for each timer, with the following capabilities (the number of match outputs available for each timer on a chip may vary):
  - Go low when matched
  - Go high when matched
  - Toggle when matched
  - Do nothing
- Enables you to configure up to four match registers for PWM operation, allowing up to three single-edged controlled PWM outputs (the number of match outputs available for each timer on a chip may vary)
- Uses up to two match registers to generate DMA requests

48.3 Functional description

Figure 253 shows a timer configured to reset the count and generate an interrupt after a match. The value of Prescale (PR) is 2 and the value of Match (MR0 - MR3) is 6. At the end of the timer cycle where the match occurs, the timer count is reset and gives a full-length cycle to the match value. The interrupt indicating that a match has occurred is generated in the next clock after the timer reaches the match value.

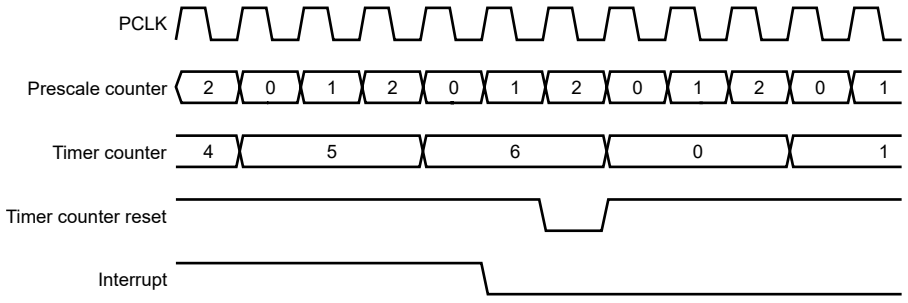


Figure 253. Timer cycle with PR = 2, MR*x* = 6, and both interrupt and reset enabled after match

Figure 254 shows a timer configured to stop and generate an interrupt after a match. The value of Prescale (PR) is 2 and the value of Match (MR0 - MR3) is 6. In the next clock after the timer reaches the match value, TCR[CEN] becomes 0 and the interrupt indicating that a match has occurred is generated.

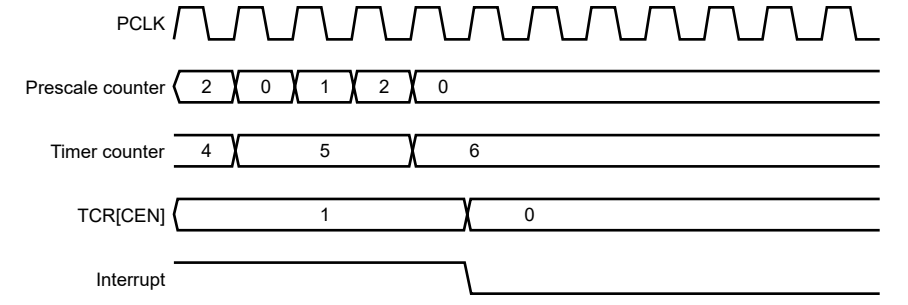


Figure 254. Timer cycle with PR = 2, MR*x* = 6, and both interrupt and stop enabled after match

48.3.1 Operation sections

All single-edge controlled PWM outputs go low at the beginning of each PWM cycle (timer is set to zero) unless their match value is zero.

Each PWM output goes high when its match value is reached. If no match occurs (that is, the match value is greater than the PWM cycle length), the PWM output remains continuously low.

If a match value larger than the PWM cycle length is written to Match (MR0 - MR3), and the PWM signal is already high, the PWM signal is cleared at the start of the next PWM cycle.

If a match register contains the same value as the timer reset value (the PWM cycle length), the PWM output is reset to low on the next clock tick after the timer reaches the match value. Therefore, the PWM output always consists of one clock tick wide positive pulse with a period that the PWM cycle length determines (that is, the timer reload value).

If Match (MR0 - MR3) = 0, the PWM output goes to high the first time the timer goes back to zero, and remains high continuously.

The following figure shows sample PWM waveforms with a PWM cycle length of 100 that MR3 specifies and PWM outputs that PWM Control (PWMC) enables.

NOTE

When you select match outputs to perform as PWM outputs, you must write 0 to MCR[MR*n*R] and MCR[MR*n*S], except to the match register setting the PWM cycle length. For this register, write 1 to MCR[MR*n*R] to enable the timer reset when the timer value matches the value of the corresponding match register.

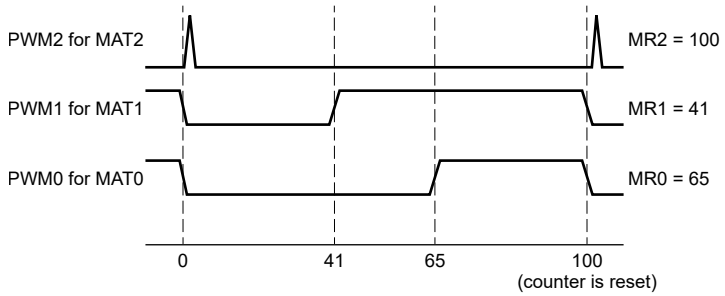


Figure 255. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWM Control register.

48.3.2 Mode sections

48.3.2.1 Capture mode

You can configure a CAP to load Capture (CR0 - CR3) with the value in the counter or timer and optionally generate an interrupt. One of the pins generates the capture signal with a capture function. Each CAP is connected to one capture channel of the timer.

CTIMER can select a CAP as a clock source instead of selecting the APB CTIMER function clock. See Count Control (CTCR) for more information.

48.3.2.2 Match mode

When the value of a match register (Match (MR0 - MR3)) equals that of Timer Counter (TC), the corresponding match output can either toggle, go low, go high, or do nothing. External Match (EMR) and PWM Control (PWMC) control the functionality of this output.

48.3.3 External trigger enable

When trigger_enable is asserted (rising edge), it is equivalent to writing 1 to TCR[CEN]. If TCR[CEN] is already 1, trigger_enable rising edge events have no effect.

48.3.4 Clocking

Table 428. CTIMER clocks

Type of clock	Description
Functional Clock(ctclk)	Asynchronous to the bus clock, provide clock to perform counter or timer operations.
Bus Clock(pclk)	The bus clock is only used for bus accesses to the control and configuration registers.

48.3.5 Reset

TCR[CRST] resets Timer Counter (TC) and Prescale Counter (PC).

48.3.6 Interrupts

This module has capture and match interrupts. See the Interrupt (IR)

48.3.7 DMA

DMA requests are generated when the value of Timer Counter (TC) either matches the value of MR0 or MR1 (see Match (MR0 - MR3) for more information). This DMA request is not connected to the operation of the match outputs (see Match mode) controlled by External Match (EMR). Each match sets a DMA request flag, which is connected to the DMA controller. You must configure the DMA controller correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by writing 1 to the interrupt flag location, as if clearing a timer interrupt. See Interrupt (IR) for more information. A DMA request is cleared automatically when the DMA controller manages it.

NOTE

Timer DMA requests are generated whenever the timer value is equal to the related match register value. DMA requests are always generated when the timer is running, unless the match register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless you configure the timer correctly to generate valid DMA requests.

48.4 External signals

Table 429. CTIMER pin descriptions

Pin	Type	Description
CTIMER*m*_CAP*n*	Input	Capture signals—You can configure a transition on a capture pin to load one of the capture registers (Capture (CR0 - CR3)) with the value of Timer Counter (TC) and optionally generate an interrupt. You can select the capture functionality for a number of pins. CTIMER can select a capture signal as a clock source instead of the APB bus clock. See Count Control (CTCR) for more information.
CTIMER*m*_MAT*n*	Output	External match output—When the value of a match register (Match (MR0 - MR3)) equals the value of Timer Counter (TC), this output can either toggle, go low, go high, or do nothing. External Match (EMR) controls the functionality of this output. You can select the match output functionality for a number of pins in parallel.

48.4.1 Multiple capture (CAP) and match (MAT) pins

You can configure the pins of the chip to be used for this module. These pins are generally reused, and you can connect several internal module pins selectively. However, you can connect only one internal pin at a time.

NOTE

Match conditions may be used internally without the use of a chip's pin.

48.5 Initialization

See the Chip-specific CTIMER information for the initialization section for more details or steps.

48.6 Application information

CTIMER register descriptions discusses the following:

- Interval timer for counting internal events
- PWM outputs
- Pulse width demodulator via capture inputs
- Free running timer

48.7 CTIMER register descriptions

48.7.1 CTIMER memory map

CTIMER0 base address: 4000_C000h

CTIMER1 base address: 4000_D000h

CTIMER2 base address: 4000_E000h

CTIMER3 base address: 4000_F000h

CTIMER4 base address: 4001_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Interrupt (IR)	32	RW	See section
4h	Timer Control (TCR)	32	RW	See section
8h	Timer Counter (TC)	32	RW	0000_0000h
Ch	Prescale (PR)	32	RW	0000_0000h
10h	Prescale Counter (PC)	32	RW	0000_0000h
14h	Match Control (MCR)	32	RW	See section
18h - 24h	Match (MR0 - MR3)	32	RW	0000_0000h
28h	Capture Control (CCR)	32	RW	See section
2Ch - 38h	Capture (CR0 - CR3)	32	R	0000_0000h
3Ch	External Match (EMR)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
70h	Count Control (CTCR)	32	RW	See section
74h	PWM Control (PWMC)	32	RW	See section
78h - 84h	Match Shadow (MSR0 - MSR3)	32	RW	0000_0000h

48.7.2 Interrupt (IR)

Offset

Register	Offset
IR	0h

Function

Provides flags for match interrupts and capture interrupts. If an interrupt is generated, it sets the corresponding flag in this register. Otherwise, the flag remains cleared. Writing 1 to the corresponding IR field resets the interrupt. Writing 0 has no effect. Clearing an interrupt for a timer match also clears any corresponding DMA request. You can write to this register to clear interrupts, and can read this register to identify which interrupt sources are pending.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								CR3INT	CR2INT	CR1INT	CR0INT	MR3INT	MR2INT	MR1INT	MR0INT
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-4 CRnINT	Interrupt Flag for Capture Channel n Event Controls an interrupt flag for capture channel *n* events.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 MRnINT	Interrupt Flag for Match Channel n Event Controls an interrupt flag for match channel *n* events.

48.7.3 Timer Control (TCR)

Offset

Register	Offset
TCR	4h

Function

Controls timer counter functions. You can disable or reset Timer Counter (TC) by using this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										ATCEN	AGCEN	Reserved		CRST	CEN
Reset	u	u	u	u	u	u	u	u	u	u	0	0	u	u	0	0

Fields

Field	Function
31-6 —	Reserved
5 ATCEN	Allow Trigger Count Enable Enables the input trigger. If this field is 1, it allows the input trigger_enable = 1 action to take effect (see External trigger enable). If this field is 0, the action is not allowed. 0b - Disable 1b - Enable
4 AGCEN	Allow Global Count Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Enables the global count. If this field is 1, it allows the input global_enable = 1 action to take effect. If this field = 0, the action is not allowed. 0b - Disable 1b - Enable
3-2 —	Reserved
1 CRST	Counter Reset Enable Enables the counter reset. If this field is 1, Timer Counter (TC) and Prescale Counter (PC) are synchronously reset after the next positive edge of the CTIMER function clock. The counters remain reset until this field becomes 0. If this field is 0, no effect takes place. 0b - Disable 1b - Enable
0 CEN	Counter Enable Enables the counters. If this field is 1, Timer Counter (TC) and Prescale Counter (PC) are enabled. When an external trigger enables the timer or the external global start enable register enables it, this field becomes 1 automatically. 0b - Disable 1b - Enable

48.7.4 Timer Counter (TC)

Offset

Register	Offset
TC	8h

Function

Specifies the timer counter.

This register is incremented when Prescale Counter (PC) reaches its terminal count. Unless it is reset before reaching its upper limit, Timer Counter (TC) counts up through the value FFFF_FFFFh, and then wraps back to the value 0000_0000h. This event does not cause an interrupt, but you can use Match (MR0 - MR3) to detect an overflow if needed.

Timer Counter (TC) is incremented after every Prescale (PR) + 1 cycle of the CTIMER function clock. Timer Control (TCR) controls the functioning of this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 TCVAL	Timer Counter Value Specifies the value of the timer counter.

48.7.5 Prescale (PR)

Offset

Register	Offset
PR	Ch

Function

Specifies the prescale value.

If Prescale Counter (PC) is equal to the prescale value, the next clock timer increments Timer Counter (TC) and turns the value of Prescale Counter (PC) to 0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PRVAL	Prescale Reload Value Specifies the value of the prescale reload.

48.7.6 Prescale Counter (PC)

Offset

Register	Offset
PC	10h

Function

Controls the division of the CTIMER function clock by some constant value before it is applied to Timer Counter (TC). This allows control of the relationship between the resolution of the timer and the maximum time before the timer overflows.

Prescale Counter (PC) is incremented on every CTIMER function clock. When it reaches the value stored in Prescale (PR), Timer Counter (TC) is incremented and Prescale Counter (PC) is reset on the next CTIMER function clock. This causes Timer Counter (TC) to increment on every CTIMER function clock when Prescale (PR) = 0, every two CTIMER function clocks when Prescale (PR) = 1, and so on.

You can use the bus interface to observe and control Prescale Counter (PC).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PCVAL	Prescale Counter Value Specifies the value of the prescale counter.

48.7.7 Match Control (MCR)

Offset

Register	Offset
MCR	14h

Function

Controls the operations that are performed when one of the match registers (Match (MR0 - MR3)) matches Timer Counter (TC).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					MR3RL	MR2RL	MR1RL	MR0RL								
W	Reserved								Reserved							
Reset	u	u	u	u	0	0	0	0	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					MR3S	MR3R	MR3I	MR2S	MR2R	MR2I	MR1S	MR1R	MR1I	MR0S	MR0R	MR0I
W	Reserved															
Reset	u	u	u	u	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 MRnRL	Reload MR Reloads with the contents of the match n shadow register (see Match Shadow (MSR0 - MSR3)) when Timer Counter (TC) is reset to zero (either via a match event or by writing 1 to TCR[CRST]). 0b - Does not reload 1b - Reloads
23-12 —	Reserved
11 MR3S	Stop on MR3 Stops Timer Counter (TC) and Prescale Counter (PC), and turns TCR[CEN] to 0 if MR3 matches Timer Counter (TC). 0b - Does not stop 1b - Stops

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 MR3R	Reset on MR3 Resets Timer Counter (TC) if MR3 matches its value. 0b - Does not reset 1b - Resets
9 MR3I	Interrupt on MR3 Generates an interrupt when MR3 matches the value in Timer Counter (TC). 0b - Does not generate 1b - Generates
8 MR2S	Stop on MR2 Stops Timer Counter (TC) and Prescale Counter (PC), and turns TCR[CEN] to 0 if MR2 matches Timer Counter (TC). 0b - Does not stop 1b - Stops
7 MR2R	Reset on MR2 Resets Timer Counter (TC) if MR2 matches its value. 0b - Does not reset 1b - Resets
6 MR2I	Interrupt on MR2 Generates an interrupt when MR2 matches the value in Timer Counter (TC). 0b - Does not generate 1b - Generates
5 MR1S	Stop on MR1 Stops Timer Counter (TC) and Prescale Counter (PC), and turns TCR[CEN] to 0 if MR1 matches Timer Counter (TC). 0b - Does not stop 1b - Stops
4 MR1R	Reset on MR1 Resets Timer Counter (TC) if MR1 matches its value. 0b - Does not reset 1b - Resets
3 MR1I	Interrupt on MR1 Generates an interrupt when MR1 matches the value in Timer Counter (TC).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Does not generate 1b - Generates
2 MR0S	Stop on MR0 Stops Timer Counter (TC) and Prescale Counter (PC), and turns TCR[CEN] to 0 if MR0 matches Timer Counter (TC). 0b - Does not stop 1b - Stops
1 MR0R	Reset on MR0 Resets Timer Counter (TC) if MR0 matches its value. 0b - Does not reset 1b - Resets
0 MR0I	Interrupt on MR0 Generates an interrupt when MR0 matches the value in Timer Counter (TC). 0b - Does not generate 1b - Generates

48.7.8 Match (MR0 - MR3)

Offset

Register	Offset
MR0	18h
MR1	1Ch
MR2	20h
MR3	24h

Function

Continuously compares the value of Match (MR0 - MR3) with the value of TC[TCVAL]. When both the values are equal, actions such as the following can be triggered automatically:

- Generation of an interrupt
- Resetting of Timer Counter (TC)
- Stopping of the timer

The settings defined in Match Control (MCR) control these actions.

If MCR[MRnRL] = 1, Match (MR0 - MR3) is automatically reloaded with the current contents of its corresponding Match Shadow (MSR0 - MSR3) whenever Timer Counter (TC) becomes 0. This transfer takes place on the same clock edge that advances Timer Counter (TC) to 0.

NOTE

Timer Counter (TC) resets in response to an occurrence of a match on MATCH being used to set the cycle counter rate. A reset can also occur if you write 1 to TCR[CRST].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MATCH	Timer Counter Match Value Triggers an action automatically when the value of this field is the same as that of TC[TCVAL].

48.7.9 Capture Control (CCR)

Offset

Register	Offset
CCR	28h

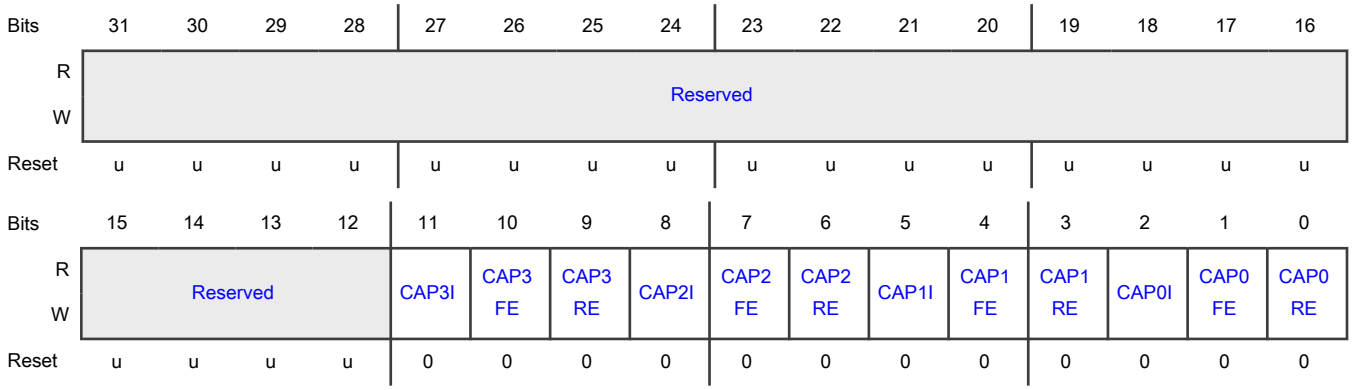
Function

Controls whether one of the capture registers (Capture (CR0 - CR3)) is loaded with the value in Timer Counter (TC) when the capture event occurs. The register also controls whether the capture event generates an interrupt. When you configure both the rising and falling bits at the same time, which is a valid configuration, it results into a capture event for both edges. In the description below, "*n*" represents the timer number, 0 or 1.

NOTE

If you select Counter mode for a particular CAP*n* input in Count Control (CTCR), you must program the fields in this register as 0. You can select both capture and interrupt for the other CAP*n* inputs.

Diagram



Fields

Field	Function
31-12 —	Reserved
11 CAP3I	Generate Interrupt on Channel 3 Capture Event Generates an interrupt on channel 3 capture event using a CR3 load. 0b - Does not generate 1b - Generates
10 CAP3FE	Falling Edge of Capture Channel 3 Loads CR3 with the contents of Timer Counter (TC) when you write 1 and then 0 to this field. 0b - Does not load 1b - Loads
9 CAP3RE	Rising Edge of Capture Channel 3 Loads CR3 with the contents of Timer Counter (TC) when you write 0 and then 1 to this field. 0b - Does not load 1b - Loads
8 CAP2I	Generate Interrupt on Channel 2 Capture Event Generates an interrupt on channel 2 capture event using a CR2 load. 0b - Does not generate 1b - Generates
7 CAP2FE	Falling Edge of Capture Channel 2 Loads CR2 with the contents of Timer Counter (TC) when you write 1 and then 0 to this field. 0b - Does not load 1b - Loads

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 CAP2RE	Rising Edge of Capture Channel 2 Loads CR2 with the contents of Timer Counter (TC) when you write 0 and then 1 to this field. 0b - Does not load 1b - Loads
5 CAP1I	Generate Interrupt on Channel 1 Capture Event Generates an interrupt on channel 1 capture event using a CR1 load. 0b - Does not generates 1b - Generates
4 CAP1FE	Falling Edge of Capture Channel 1 Loads CR1 with the contents of Timer Counter (TC) when you write 1 and then 0 to this field. 0b - Does not load 1b - Loads
3 CAP1RE	Rising Edge of Capture Channel 1 Loads CR1 with the contents of Timer Counter (TC) when you write 0 and then 1 to this field. 0b - Does not load 1b - Loads
2 CAP0I	Generate Interrupt on Channel 0 Capture Event Generates an interrupt on channel 0 capture event using a CR0 load. 0b - Does not generate 1b - Generates
1 CAP0FE	Falling Edge of Capture Channel 0 Loads CR0 with the contents of Timer Counter (TC) when you write 1 and then 0 to this field. 0b - Does not load 1b - Loads
0 CAP0RE	Rising Edge of Capture Channel 0 Loads CR0 with the contents of Timer Counter (TC) when you write 0 and then 1 to this field. 0b - Does not load 1b - Loads

48.7.10 Capture (CR0 - CR3)

Offset

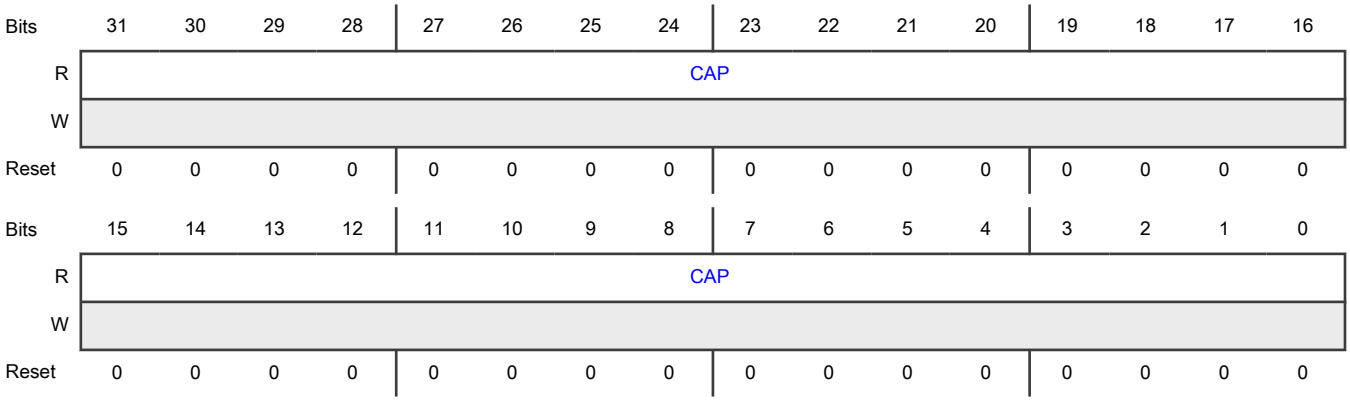
Register	Offset
CR0	2Ch
CR1	30h
CR2	34h
CR3	38h

Function

Works with a capture channel and may load with the value of Timer Counter (TC) when a specified event occurs on the signal defined for that capture channel. The signal could originate from an external pin or an internal source.

This register determines whether the capture function is enabled, and whether a capture event happens on the rising edge, falling edge, or on both edges of the associated signal. The register is loaded with the value of Timer Counter (TC) when there is an event on the CAP*n* input.

Diagram



Fields

Field	Function
31-0 CAP	Timer Counter Capture Value Determines whether the capture function is enabled and defines the edge on which the capture function occurs.

48.7.11 External Match (EMR)

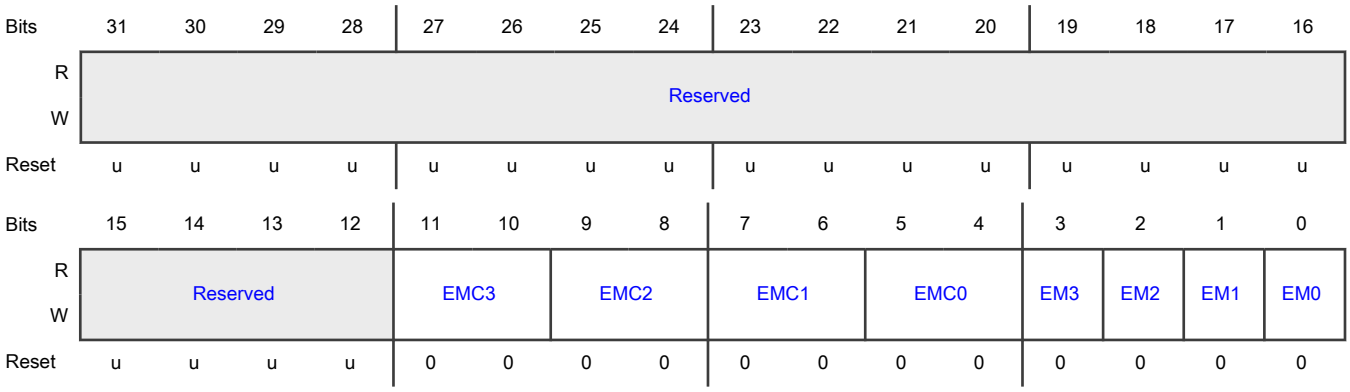
Offset

Register	Offset
EMR	3Ch

Function

Provides both control and status of the external match pins. If the match outputs (see Match mode) are configured as PWM outputs, PWM rules determine the function of this register.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-10 EMC3	External Match Control 3 Determines the functionality of EM3: • If this field is 1, the corresponding external match field or output becomes 0 (if pinned out MAT3 pin is low). • If this field is 10, the corresponding external match field or output becomes 1 (if pinned out MAT3 pin is high). • If this field is 11, it toggles the corresponding external match field or output. 00b - Does nothing 01b - Goes low 10b - Goes high 11b - Toggles
9-8 EMC2	External Match Control 2 Determines the functionality of EM2: • If this field is 1, the corresponding external match field or output becomes 0 (if pinned out MAT2 pin is low). • If this field is 10, the corresponding external match field or output becomes 1 (if pinned out MAT2 pin is high). • If this field is 11, it toggles the corresponding external match field or output. 00b - Does nothing

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Goes low 10b - Goes high 11b - Toggles
7-6 EMC1	External Match Control 1 Determines the functionality of EM1: • If this field is 1, the corresponding external match field or output becomes 0 (if pinned out MAT1 pin is low). • If this field is 10, the corresponding external match field or output becomes 1 (if pinned out MAT1 pin is high). • If this field is 11, it toggles the corresponding external match field or output. 00b - Does nothing 01b - Goes low 10b - Goes high 11b - Toggles
5-4 EMC0	External Match Control 0 Determines the functionality of EM0: • If this field is 1, the corresponding external match field or output becomes 0 (if pinned out MAT0 pin is low). • If this field is 10, the corresponding external match field or output becomes 1 (if pinned out MAT0 pin is high). • If this field is 11, it toggles the corresponding external match field or output. 00b - Does nothing 01b - Goes low 10b - Goes high 11b - Toggles
3 EM3	External Match 3 Reflects the state of output MAT3—whether this output is connected to a pin. If a match occurs between Timer Counter (TC) and MR3 (see Match (MR0 - MR3)), this field can either toggle, go low, go high, or do nothing, as selected by EMC3. This field is driven to the MAT pins if the match function is selected via IOPCTL. 0b - Low 1b - High
2 EM2	External Match 2 Reflects the state of output MAT2—whether this output is connected to a pin. If a match occurs between Timer Counter (TC) and MR2 (see Match (MR0 - MR3)), this field can either toggle, go low, go high, or

Table continues on the next page...

Table continued from the previous page...

Field	Function
	do nothing, as selected by EMC2. This field is driven to the MAT pins if the match function is selected via IOPCTL. 0b - Low 1b - High
1 EM1	External Match 1 Reflects the state of output MAT1—whether this output is connected to a pin. If a match occurs between Timer Counter (TC) and MR1 (see Match (MR0 - MR3)), this field can either toggle, go low, go high, or do nothing, as selected by EMC1. This field is driven to the MAT pins if the match function is selected via IOPCTL. 0b - Low 1b - High
0 EM0	External Match 0 Reflects the state of output MAT0—whether this output is connected to a pin. If a match occurs between Timer Counter (TC) and MR0 (see Match (MR0 - MR3)), this field can either toggle, go low, go high, or do nothing, as selected by EMC0. This field is driven to the MAT pins if the match function is selected via IOPCTL. 0b - Low 1b - High

48.7.12 Count Control (CTCR)

Offset

Register	Offset
CTCR	70h

Function

Selects between Timer and Counter mode, and in Counter mode, selects the signal and edges for counting.

When Counter mode is selected as the mode of operation, the CAP input (selected by CINSEL) is sampled on every rising edge of the CTIMER function clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized:

- Rising edge
- Falling edge
- Either of the edges
- No change in the selected CAP input level

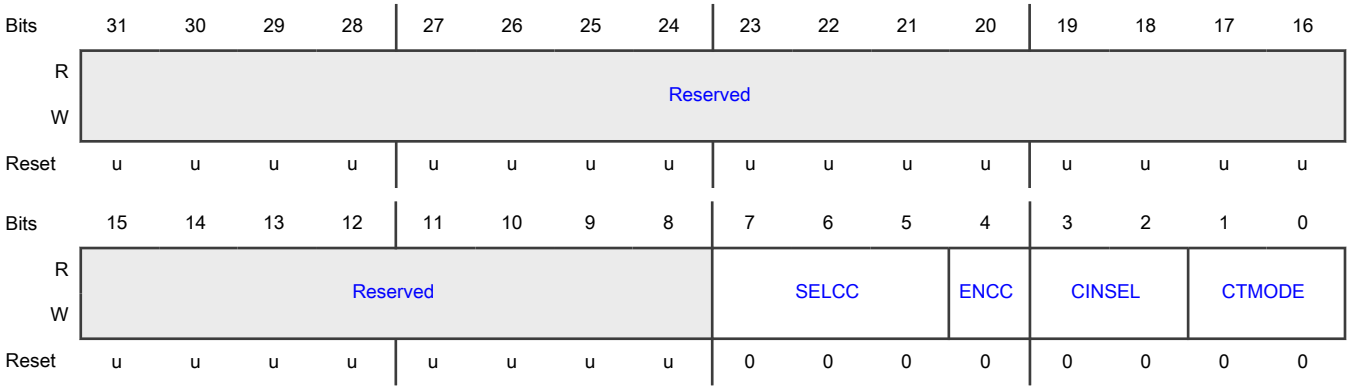
Only if the identified event occurs and the event corresponds to the one selected by CTMODE, Timer Counter (TC) is incremented.

Effective processing of the externally supplied clock to the counter has some limitations:

- Because two successive rising edges of the CTIMER function clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the CTIMER function clock.
- Consequently, the duration of the high or low levels on the same CAP input in this case cannot be shorter than the CTIMER function clock duration.

You can also use ENCC and SELCC to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and perform a capture on the trailing edge permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-5 SELCC	Edge Select Selects which input edge (rising or falling) causes the timer and prescaler to be cleared, when ENCC is 1. These fields have no effect when ENCC is 0. 000b - Capture channel 0 rising edge 001b - Capture channel 0 falling edge 010b - Capture channel 1 rising edge 011b - Capture channel 1 falling edge 100b - Capture channel 2 rising edge 101b - Capture channel 2 falling edge 110b - Capture channel 3 rising edge 111b - Capture channel 3 falling edge
4 ENCC	Capture Channel Enable Enables capture channel. Writing 1 to this field enables clearing of the timer and the prescaler when the capture-edge event specified in SELCC occurs.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 CINSEL	Count Input Select Selects which CAP pin is sampled for clocking when CTMODE is not 0. **NOTE** If Counter mode is selected for a particular CAP*n* input in Count Control (CTCR), you must program the corresponding fields for that input in Capture Control (CCR) as 0. However, you can select both capture and interrupt for the other three CAP*n* inputs in the same timer. 00b - Channel 0, CAPn[0] for CTIMERn 01b - Channel 1, CAPn[1] for CTIMERn 10b - Channel 2, CAPn[2] for CTIMERn 11b - Channel 3, CAPn[3] for CTIMERn
1-0 CTMODE	Counter Timer Mode Selects between Timer and Counter mode, and in Counter mode, selects the signal and edges for counting: • If this field is 0, every rising CTIMER function clock edge is incremented. • If this field is 1, Timer Counter (TC) is incremented on the rising edges of the CAP input selected by CINSEL. • If this field is 10, Timer Counter (TC) is incremented on the falling edges of the CAP input selected by CINSEL. • If this field is 11, Timer Counter (TC) is incremented on both edges of the CAP input selected by CINSEL. 00b - Timer mode 01b - Counter mode rising edge 10b - Counter mode falling edge 11b - Counter mode dual edge

48.7.13 PWM Control (PWMC)

Offset

Register	Offset
PWMC	74h

Function

Configures match outputs (see Match mode) as PWM outputs. You can independently set each match output to perform either as a PWM output or match output controlled by External Match (EMR).

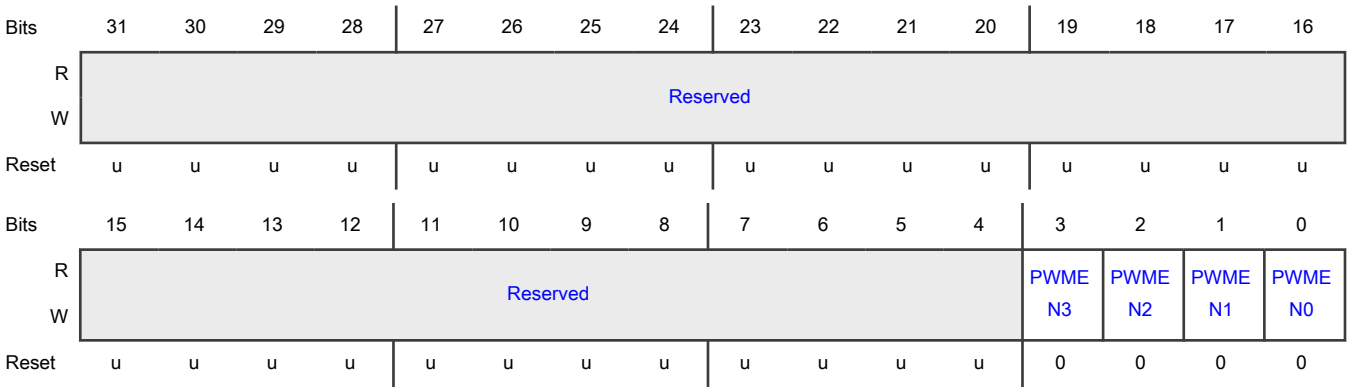
NOTE

Different part numbers and package variations may provide different match output pin functions.

For each timer, you can select a maximum of three single-edge controlled PWM outputs on the MAT*n*[2:0] outputs. One additional match register (see Match (MR0 - MR3)) determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to high. The match register resets the timer that you can configure to set the PWM cycle length. When the timer is reset to zero, all currently high match outputs configured as PWM outputs are cleared.

This register enables PWM mode for the external match pins.

Diagram



Fields

Field	Function
31-4 —	Reserved
3 PWMEN3	PWM Mode Enable for Channel 3 Enables PWM mode for channel 3. Use match channel 3 to set the PWM cycle length. If this field is 0, EM3 controls CTIMER*n*_MAT3. If this field is 1, PWM mode is enabled for CTIMER*n*_MAT3. 0b - Disable 1b - Enable
2 PWMEN2	PWM Mode Enable for Channel 2 Enables PWM mode for channel 2. If this field is 0, EM2 controls CTIMER*n*_MAT2. If this field is 1, PWM mode is enabled for CTIMER*n*_MAT2. 0b - Disable 1b - Enable
1 PWMEN1	PWM Mode Enable for Channel 1 Enables PWM mode for channel 1. If this field is 0, EM1 controls CTIMER*n*_MAT1. If this field is 1, PWM mode is enabled for CTIMER*n*_MAT1. 0b - Disable 1b - Enable
0	PWM Mode Enable for Channel 0

Table continues on the next page...

Table continued from the previous page...

Field	Function
PWMEN0	Enables PWM mode for channel 0. If this field is 0, EM0 controls CTIMER*n*_MAT0. If this field is 1, PWM mode is enabled for CTIMER*n*_MAT0. 0b - Disable 1b - Enable

48.7.14 Match Shadow (MSR0 - MSR3)

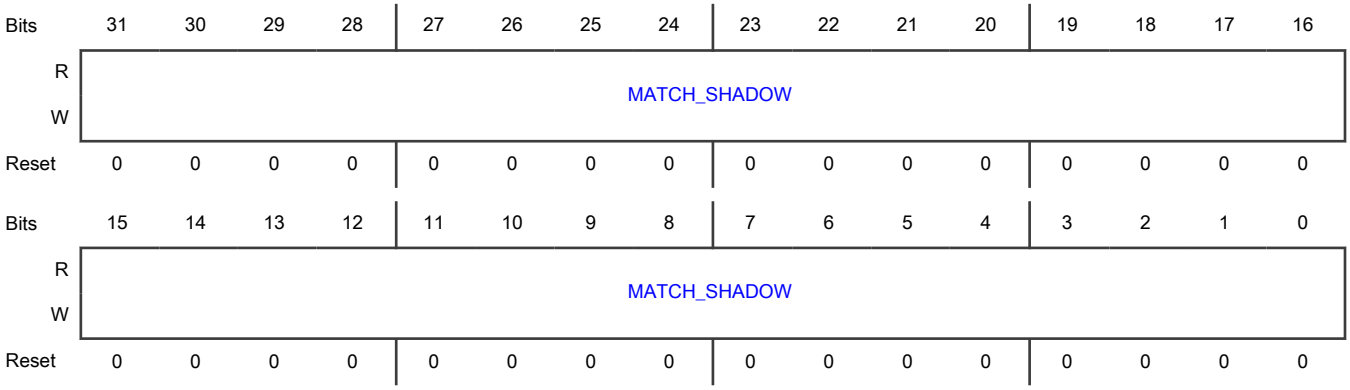
Offset

Register	Offset
MSR0	78h
MSR1	7Ch
MSR2	80h
MSR3	84h

Function

Contains the value that the corresponding Match (MR0 - MR3) are (optionally) reloaded with at the start of each new counter cycle. Typically, the match that causes the counter to be reset (and instigates the match reload) is also programmed to generate an interrupt or DMA request. Software or the DMA engine then has one full counter cycle to modify the contents of this register before the next reload occurs.

Diagram



Fields

Field	Function
31-0 MATCH_SHADOW	Timer Counter Match Shadow Value Reloads automatically with the contents of this register whenever Timer Counter (TC) is reset to zero.

Chapter 49
SCTIMER

49.1 Chip-specific SCTIMER information

Table 430. Reference links to related information

Topic	Related module	Reference
Full description	SCTIMER	SCTIMER
Peripheral memory map		PBRG Memory map
Interrupts	NVIC	Interrupts
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See SCT0_INMUX0 - SCT0_INMUX7 registers in INPUTMUX

49.1.1 Module instances

This device has one instance of the SCTIMER module, SCT0.

49.2 Overview

SCT is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention. It can operate as a single 32-bit counter or as two independent, 16-bit counters in Unidirectional or Bidirectional modes.

As with most timers, SCT supports:

- A selection of match registers against which you can compare the count value.
- Capture registers where you can record the current count value if a predefined condition is detected.

An additional feature contributing to the versatility of SCT is the concept of events. This module supports multiple separate events that you can define based on some combination of parameters, including all or some of the following:

- A match on one of the match registers
- A transition on one of the SCT inputs or outputs
- The direction of count

Every action that SCT performs occurs in direct response to the events that you define, without any software overhead. You can enable an event to:

- Start, stop, or halt the counter.
- Limit the counter, which refers to clearing the counter in Unidirectional mode, or changing the counter direction in Bidirectional mode.
- Set, clear, or toggle any SCT output.
- Force a capture of the count value into any of the capture registers.
- Generate an interrupt or DMA request.

SCT also allows you to group and filter events. A group of enabled and disabled events can be described as a state, and several states with different sets of enabled and disabled events are allowed. Changing from one state to another is event-driven and can therefore happen without your intervention. By defining these states, SCT provides the means to run state machines in hardware to accomplish complex waveform and timing tasks.

In a simple system such as a classical timer or counter with capture and match capabilities, all events that could cause the timer to capture the timer value or toggle a match output are enabled, and remain enabled, at all times while the counter is running. In this case, no events are filtered and the state that does not change describes the system. This is the default configuration for SCT.

In a more complex system, two states could be set up that allow some events in one state and not in the other. You can use an event that is enabled in both states to move from one state to the other and back while filtering out events in either of the states. In such a two-state system, you can create different waveforms at the SCT output, depending on the event history. Change of states is event-driven and happens without any CPU intervention.

Formally, you can program SCT as a state machine generator. With the unique ability to perform switching between groups of events, you can use SCT as a highly complex state machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides you with an extremely powerful control tool—particularly when SCT inputs and outputs are connected to other on-chip resources.

In addition to events and states, SCT provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode
- Selection of any SCT input as a clock source or a clock gate
- Ability to define a fractional portion to certain match registers (fractional matching uses a “dither” approach to achieve average resolutions of up to 1/16th of a clock period when generating PWM edges)
- Ability to select a greater-than-or-equal-to match condition for the purpose of event generation

NOTE

In this chapter, the term "bus error" indicates an SCT response that makes the processor take an exception.

49.2.1 Block diagram

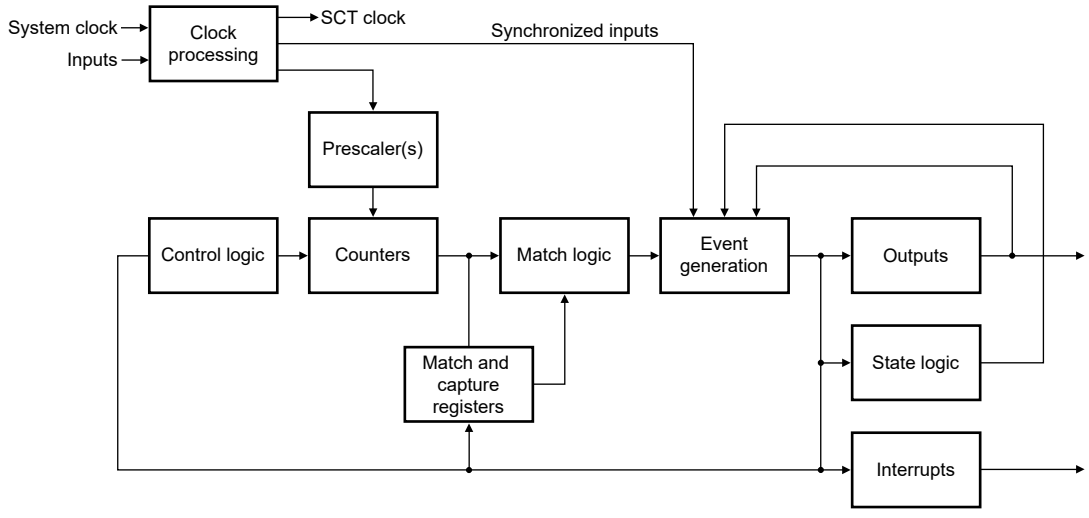


Figure 256. Block diagram

49.2.2 Features

SCT supports:

- 8 inputs

- 10 outputs
- 16 match and capture registers
- 16 events
- 32 states
- 6 dithers

Counter or timer features:

- Configurable as two 16-bit counters or one 32-bit counter (this configuration is possible for each SCT)
- Counters clocked by the system clock or a selected input
- Configurable as up counters or up-down counters
- Configurable number of match and capture registers
- Ability for the counter value to be loaded into capture registers, triggered by a match or I/O toggle
- Event creation possible for the following timer events for match and/or I/O transitions:
  - DMA request
  - Interrupt
  - Stop
  - Limit
  - Halt
  - Change counting direction
  - Toggle outputs
  - Change the state

PWM features:

- Ability to use counters with match registers to toggle outputs and create time-proportioned PWM signals (PWM waveforms can change based on the current state)
- Up to eight single-edge or four dual-edge PWM outputs with an independent duty cycle and a common PWM cycle length

Event creation features:

- Ability to enable events in Bidirectional mode, based on the count direction (the following conditions define an event):
  - A counter match condition
  - An input (or output) condition such as a rising or falling edge or level
  - A combination of match and/or I/O conditions
- Ability for selected events to limit, halt, start, or stop a counter or change direction
- Ability for events to trigger state changes, output transitions, timer captures, interrupts, and DMA transactions
- Ability to use match register 0 as an automatic limit
- Ability to hold match events until another qualifying event occurs

State control features:

- No pre-defined meaning (you determine these states entirely)
- Provision for a mechanism of context switching for SCT, including creation of complex state machines
- Defined by events that can happen in the state while the counter is running
- Ability to transition into another state as a result of an event

- Ability to be assigned to one or more states
- Ability for state variables to allow sequencing across multiple counter cycles

49.3 Functional description

49.3.1 SCT operation

In its simplest, single-state configuration, SCT operates as an event-controlled single or bidirectional counter. You can configure events as:

- Counter match events
- An input or output level
- Transitions on an input or output pin
- Combination of match and I/O behavior

In response to an event, SCT output (or outputs) can transition, or SCT can perform other actions such as creating an interrupt, and starting, stopping, or resetting the counter. You can perform multiple, simultaneous actions for each event. Furthermore, any number of events can trigger a specific SCT action.

A single or multiple SCT actions uniquely define an event. Events that you enable to trigger an SCT action or actions at any stage of the counter define a state, and the events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that SCT can perform when an event occurs. If you configure an event to change the state, the new state defines a new set of events resulting in different SCT actions. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event-controlled transitions on the SCT outputs and/or interrupts.

After configuration, SCT can run continuously, without your intervention, and can generate multiple output patterns entirely under the control of events. See the following sections to perform various SCT actions:

- Configuring SCT for multiple events and states to configure SCT
- Running SCT to start, run, and stop SCT
- Configuring SCT without using states to configure SCT as a simple, event-controlled counter or timer

49.3.2 Configuring SCT without using states

SCT can function as a standard counter or timer with external capture inputs and match outputs, without using the state logic. To use SCT without states, configure SCT as follows:

1. Write 0 to State Variable (STATE) (0 is the default).
2. Write 0 to EV*n*_CTRL[STATELD] and EV*n*_CTRL[STATEV] for each event.
3. Write 0x1 to EV*n*_STATE[STATEMSKn] for each event. This enables the event.

In effect, the event occurs in a single state that never changes when the counter is running.

49.3.3 Configuring SCT for multiple events and states

49.3.3.1 Configure the counter

1. Configure the low (L) and high (H) counters in SCT Configuration (CONFIG) by selecting two independent 16-bit counters (L and H) or one combined 32-bit counter in CONFIG[UNIFY].
2. Select the SCT clock source by using CONFIG[CLKMODE] and CONFIG[CKSEL] from any of the inputs or an internal clock.

49.3.3.2 Configure the match and capture registers

1. Select the number of match and capture registers that the application uses (not more than what is available on this chip), by using Match and Capture Register Mode (REGMODE). For each of the match and capture register pairs, you can select whether the register is used as a match register or capture register.
2. Define match conditions for each selected match register:
   - Each match register (Match Value (MATCH0 - MATCH15)) sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
   - Each match reload register (Match Reload Value (MATCHREL0 - MATCHREL15)) sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

49.3.3.3 Configure events and event responses

State variables allow SCT control across more than one cycle of the counter. Counter matches, I/O edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values (see Figure 257). Perform this procedure to configure events and event responses:

1. Define what triggers each event, by using Event n Control (EV0_CTRL - EV15_CTRL) (one register per event):
   - Using EV*n*_CTRL[COMBMODE], select whether the event occurs:
     - On change of an input or output
     - On an input or output level
     - On a match condition of the counter
     - On a combination of match and I/O conditions
   - For a match condition:
     a. Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in EV*n*_CTRL[MATCHSEL].
     b. Define, by using EV*n*_CTRL[HEVENT], whether the event occurs on matching the L or the H counter (when using L and H counters).
   - For an SCT input or output level or transition:
     a. Select the input number or the output number that is associated with this event, using EV*n*_CTRL[IOSEL] and EV*n*_CTRL[OUTSEL].
     b. Define how the selected input or output triggers the event (edge- or level-sensitive) using EV*n*_CTRL[IOCOND].
2. Define the effect of each event on SCT outputs, by using Output n Set (OUT0_SET - OUT9_SET) or Output n Clear (OUT0_CLR - OUT9_CLR) (up to the maximum number of outputs on this chip, one register per output): For each SCT output, select the events that set or clear this output. More than one event can change the output, and each event can change multiple outputs.
3. Define how each event affects the counter:
   a. Write 1 to the corresponding LIMIT[LIMMSK_H] or LIMIT[LIMMSK_L] for the event to set an upper limit for the counter:
      - When a limit event occurs in Unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.
      - When a limit event occurs in Bidirectional mode, the counter begins to count down from the current value on the next clock edge.
   b. Write 1 to the corresponding HALT[HALTMSK_H] or HALT[HALTMSK_L] for the event to halt the counter. If the counter is halted, it stops counting, and no new events can occur. You can restore the counter operation by writing 0 CTRL[HALT_H] and/or CTRL[HALT_L].

c. Write 1 to the corresponding STOP[STOPMSK_H] or STOP[STOPMSK_L] for the event to stop the counter. If the counter is stopped, it stops counting. However, if you configure an event as a transition on an I/O, it can restart the counter.

d. Write 1 to the corresponding START[STARTMSK_H] or START[STARTMSK_L] for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.

4. Define the events that contribute to the SCT interrupt: write 1 to the corresponding EVEN[IEN*n*] and EVFLAG[FLAG*n*] to enable the event to contribute to the SCT interrupt.

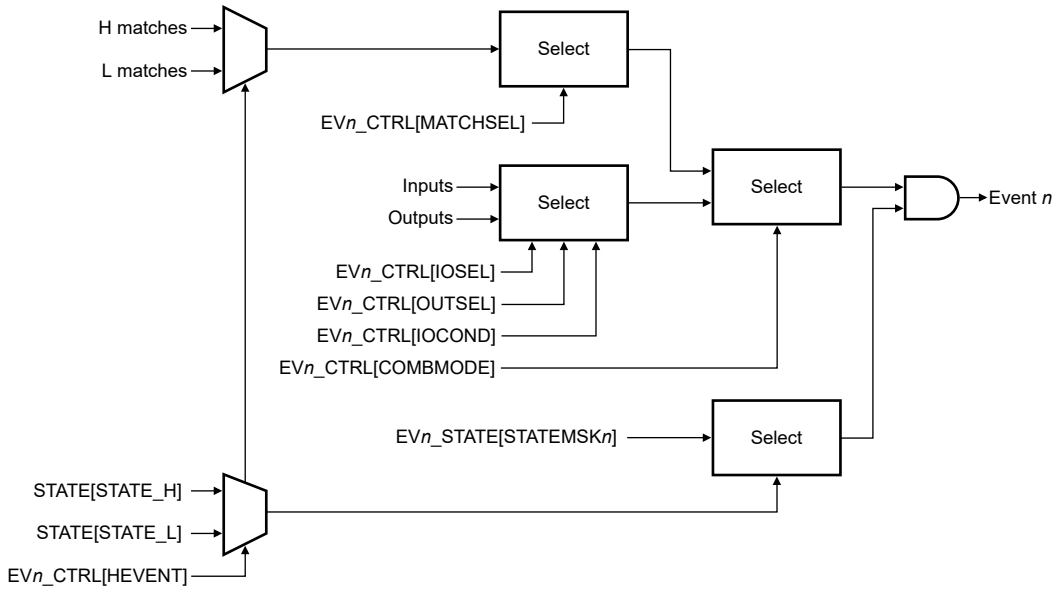


Figure 257. Event selection

49.3.3.4 Configure multiple states

1. Select the state or states (up to two), by using Event n State (EV0_STATE - EV15_STATE), for each event (up to the maximum number of events on this chip—one register per event), in which this event is allowed to occur. You can select each state for more than one event.
2. Determine how the event affects the system state: using Event n Control (EV0_CTRL - EV15_CTRL) (up to the maximum number of events on this chip—one register per event), set the new state value in EV*n*_CTRL[STATEV] for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces it, depending on the configuration of EV*n*_CTRL[STATELD].

NOTE

If EV*n*_CTRL[STATEV] and EV*n*_CTRL[STATELD] are 0, the state does not change. If there are higher numbered events in the current state, this event cannot change the state.

49.3.3.5 Miscellaneous options

SCT includes a certain (selectable) number of capture registers that you can program to capture the counter contents when one or more events occur.

If the counter is in Bidirectional mode, you can configure Output Counter Direction Control (OUTPUTDIRCTRL) to determine the effect of setting and clearing of an output, based on whether the counter is counting up or down.

49.3.4 State logic

You can configure SCT as a timer or counter with multiple programmable states, which you can define through the events that can be captured in each particular state. A multi-state SCT can change from one state to another when an event that you define triggers a state change through each event's Event n Control (EV0_CTRL - EV15_CTRL) in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, SCT enters a locked state where all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when an event that moves SCT into a locked state, occurs. Later, when SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number.

If SCT registers an event that forces an abort, putting SCT in a locked state can be a safe way to record the time that has passed since the abort event when no new events are allowed to occur. Because multiple states (any state number between the maximum implemented state and 31) are locked states, you can define multiple abort or error events, each incrementing the state number by a different value.

49.3.5 Clearing the prescaler

When enabled by a nonzero CTRL[PRE_L] and/or CTRL[PRE_H], the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- When you write to Counter Value (COUNT)
- When you write 1 to CTRL[CLRCTR]
- An event selected by a 1 in the counter limit register when CTRL[BIDIR] = 0

When CTRL[BIDIR] = 0, a limit event caused by an I/O signal can clear a nonzero prescaler. However, a limit event caused by a match clears only a nonzero prescaler in a special case, as described in Match vs. I/O events.

A limit event, when CTRL[BIDIR] = 1, does not clear the prescaler. Instead, it writes 0 to CTRL[DOWN] and decrements the counter on the same clock if the counter is enabled in that clock.

49.3.6 Match vs. I/O events

The prescaler and clock mode 01 make the counter operation complex, in which the SCT clock is the bus clock. However, you can enable the prescaler and counter to count only when a selected edge is detected on a clock input:

- The prescaler is enabled when the clock mode is not 01, or when the input edge that CONFIG[CKSEL] selects is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM = 0 or the prescaler is equal to the value of PRELIM).

An I/O component of an event can occur in any SCT clock when its counter, CTRL[HALT], is 0. In general, a match component of an event can only occur in an SCT clock when its counter fields, CTRL[HALT] and CTRL[STOP], are both 0 and the counter is enabled.

The following table shows the conditions when various kinds of events can occur.

Table 431. Event conditions

EV*n*_CTRL[COMBMODE]	EV*n*_CTRL[IOCOND]	Event can occur on clock:
I/O	Any	Whenever CTRL[HALT] = 0.

Table continues on the next page...

Table 431. Event conditions (continued)

EV*n*_CTRL[COMBMODE]	EV*n*_CTRL[IOCOND]	Event can occur on clock:
MATCH	Any	When CTRL[HALT] = 0, CTRL[STOP] = 0, and the counter is enabled.
OR	Any	• From the I/O component, whenever CTRL[HALT] = 0. • From the match component, when CTRL[HALT] = 0, CTRL[STOP] = 0, and the counter is enabled.
AND	Low or High	When CTRL[HALT] = 0, CTRL[STOP] = 0, and the counter is enabled.
AND	Rise or Fall	Whenever CTRL[HALT] = 0.

49.3.7 Fractional matches

You can configure the first six match registers from Match Value (MATCH0 - MATCH15) to have a fractional portion to their match values. A higher average resolution is achievable on these registers, with an associated Fractional Match (FRACMAT0 - FRACMAT5), by using a dithering mechanism. The dither engine delays the assertion of a match by one counter clock every *n* (0 to 15) out of 16 counter cycles. The value of *n* is specified in Fractional Match (FRACMAT0 - FRACMAT5), associated with each of the first six match registers.

You can disable dithering on any of the match registers by loading all zeroes (the default value) into its corresponding Fractional Match (FRACMAT0 - FRACMAT5).

49.3.7.1 Dithering

At the start of each new SCT counter cycle (that is, when the counter counts down to zero in Bidirectional mode or is cleared to zero by a limit event), the dither engine determines which matches are to be delayed by one clock during the upcoming counter cycle. Delaying the match effectively adds 1 to the designated match value when up-counting or subtracts 1 when down-counting, during that particular counter cycle.

For each dither-enabled match register, the value that you program in its associated Fractional Match (FRACMAT0 - FRACMAT5) specifies for how many counter cycles, out of every 16 of them, its match is to be delayed. An algorithm applied to the FRACMAT value distributes this number as evenly as possible across the 16 counter cycles. This results in a unique dither pattern for each match register (see Table 432).

Dither Condition (DITHER) (event-mask register) provides you additional control over the dithering process. Typically, the dither engine advances though the match dither patterns at the start of every new SCT counter cycle. Dither Condition (DITHER) allows you to specify that advancement to the next element in the dither patterns occurs only if one or more designated events have happened during the previous cycle of the counter.

The dither algorithm is designed to spread out the cycles in which the matches are delayed as evenly as possible across the 16 counter cycles. The following table shows the dither pattern that is applied for each value of FRACMAT. "D" indicates the counter cycles where a match on the relevant match register is delayed.

Table 432. Dither pattern

	Counter cycle															
FRACMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0h	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1h	-	-	-	-	-	-	-	-	D	-	-	-	-	-	-	-
2h	-	-	-	-	D	-	-	-	-	-	-	-	D	-	-	-

Table continues on the next page...

Table 432. Dither pattern (continued)

	Counter cycle															
FRACMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3h	-	-	-	-	D	-	-	-	D	-	-	-	D	-	-	-
4h	-	-	D	-	-	-	D	-	-	-	D	-	-	-	D	-
5h	-	-	D	-	-	-	D	-	D	-	D	-	-	-	D	-
6h	-	-	D	-	D	-	D	-	-	-	D	-	D	-	D	-
7h	-	-	D	-	D	-	D	-	D	-	D	-	D	-	D	-
8h	-	D	-	D	-	D	-	D	-	D	-	D	-	D	-	D
9h	-	D	-	D	-	D	-	D	D	D	-	D	-	D	-	D
Ah	-	D	-	D	D	D	-	D	-	D	-	D	D	D	-	D
Bh	-	D	-	D	D	D	-	D	D	D	-	D	D	D	-	D
Ch	-	D	D	D	-	D	D	D	-	D	D	D	-	D	D	D
Dh	-	D	D	D	-	D	D	D	D	D	D	D	-	D	D	D
Eh	-	D	D	D	D	D	D	D	-	D	D	D	D	D	D	D
Fh	-	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

49.4 External signals

SCT inputs can come from numerous places and are selected by the SCT input mux.

SCT outputs can be routed to multiple places and can be connected to both a pin and an ADC trigger at the same time.

Table 433. SCT signals

Signal	I/O	Description
SCT_IN*n*	I	SCT input. Signals come from GPIO sources, each from a variety of pins selected via IOPCTL muxing.
SCT_OUT*n*	O	SCT output. Signals go to IOPCTL muxing where they can be (optionally) output onto a selection of GPIO pins.

49.5 Initialization

To initialize SCT:

1. Enable the clock to SCT.
2. Clear the SCT peripheral reset.
3. Configure the interrupts.
4. Connect the SCT outputs to external pins.
5. Configure the DMA request lines to connect to the DMA trigger inputs.

49.6 Application information

49.6.1 Running SCT

1. Initialize SCT (see Initialization).
2. Write to State Variable (STATE) to define the initial state. By default, the initial state is 0.
3. To start SCT, write to SCT Control (CTRL):
    a. Clear the counters.
    b. Write 0 or 1 to CTRL[STOP_L] and/or CTRL[STOP_H].

    NOTE

    The counter starts counting after CTRL[STOP] becomes 0 as well. If CTRL[STOP] = 1, SCT, instead, waits for an event to occur that you can configure to start the counter.

    c. Select Unidirectional or Bidirectional counting mode (CTRL[BIDIR_L] and/or CTRL[BIDIR_H]) for each counter.
    d. Select the prescale factor for the counter clock (CTRL[PRE_L] or CTRL[PRE_H]).
    e. Write 0 to CTRL[HALT_L] and/or CTRL[HALT_H]. By default, the counters halt and no events can occur.
4. Stop or halt the counters at any time by using CTRL[STOP_L] and/or CTRL[STOP_H], or CTRL[HALT_L] and/or CTRL[HALT_H], respectively.
    - After the counters stop, both you and an event configured to write 0 to CTRL[STOP_L] and/or CTRL[STOP_H] can start the counter again.
    - After the counter halts, only writing 0 to CTRL[HALT_L] and/or CTRL[HALT_H] can start the counter again. No events can occur.
    - After the counters halt, you can set any SCT output as high or low by writing directly to the OUT register.

You can read the current SCT state at any time by using State Variable (STATE).

To change the current state (that is, independently of any event occurring), write 1 to CTRL[HALT] and write to State Variable (STATE) to change the state value. You are allowed to write to State Variable (STATE) only when the counter halts (CTRL[HALT_L] and/or CTRL[HALT_H] = 1) and no events can occur.

49.6.2 SCT configuration example

This section discusses SCT configuration by using an example, showing the simple application of SCT with two sets of match events (EV0/1 and EV3/4) to set and clear SCT output 0. The timer automatically resets whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to high and match event EV1 clears output 0.

When monitoring SCT input 0:

- If input 0 is low at the next timer reset (EV2), the state changes to state 1, enabling events EV3 and EV4. This leads to the same output but with different match value triggers.
- If input 0 is high at the next timer reset, the associated event (EV5) causes the state to change back to the state enabling events EV0 and EV1.

Figure 258 uses this SCT configuration:

- One input
- One output
- Five match registers
- Six events, and match 0 used with the autolimit function
- Two states

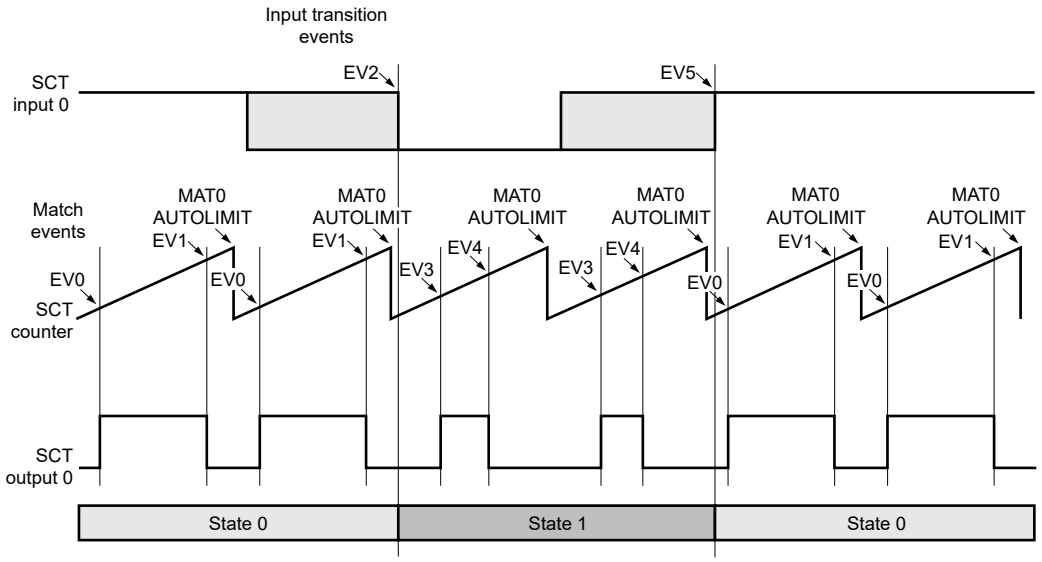


Figure 258. SCT configuration example: pictorial representation

This application of SCT uses the configuration listed in the following table (all register values not listed are their default values):

Table 434. SCT configuration example: settings

Configuration	Register	Setting
Counter	SCT Configuration (CONFIG)	Use a unified counter (CONFIG[UNIFY] = 1).
	SCT Configuration (CONFIG)	Enable the autolimit for MAT0 (CONFIG[AUTOLIMIT_L] = 1).
	SCT Control (CTRL)	Use a unidirectional counter (CTRL[BIDIR_L] = 0).
Clock base	SCT Configuration (CONFIG)	Use the default values for clock configuration.
Match and capture registers	Match and Capture Register Mode (REGMODE)	Configure one match register for each match event by writing 0 to bits 0, 1, 2, 3, and 4 of REGMODE[REGMOD_L*n*].
Define match values	MATCH (0/1/2/3/4)	Set a match value MATCH0/1/2/4_L in each register. The match register 0 serves as an automatic limit event that resets the counter, without using an event. To enable the automatic limit, write 1 to CONFIG[AUTOLIMIT].

Table continues on the next page...

Table 434. SCT configuration example: settings (continued)

Configuration	Register	Setting
Define match reload values	MATCHREL (0/1/2/3/4)	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EV0_CTRL	• Set EV*n*_CTRL[COMBMODE] = 0x1. Event 0 uses match condition only. • Set EV*n*_CTRL[MATCHSEL] = 1. Select the match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	• Set EV*n*_CTRL[COMBMODE] = 0x1. Event 1 uses match condition only. • Set EV*n*_CTRL[MATCHSEL] = 2. Select the match value of match register 2. The match value of MAT2 is associated with event 1.
Define when event 2 occurs	EV2_CTRL	• Set EV*n*_CTRL[COMBMODE] = 0x3. Event 2 uses match condition and I/O condition. • Set EV*n*_CTRL[IOSEL] = 0. Select input 0. • Set EV*n*_CTRL[IOCOND] = 0x0. Input 0 is low. • Set EV*n*_CTRL[MATCHSEL] = 0. This selects match register 0 to qualify the event = 0.
Define how event 2 changes the state	EV2_CTRL	Write 1 to EV*n*_CTRL[STATEV] and EV*n*_CTRL[STATELD]. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	• Set EV*n*_CTRL[COMBMODE] = 0x1. Event 3 uses match condition only. • Set EV*n*_CTRL[MATCHSEL] = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3.
Define when event 4 occurs	EV4_CTRL	• Set EV*n*_CTRL[COMBMODE] = 0x1. Event 4 uses match condition only. • Set EV*n*_CTRL[MATCHSEL] = 0x4. Select match value of match register 4. The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	• Set EV*n*_CTRL[COMBMODE] = 0x3. Event 5 uses match condition and I/O condition. • Set EV*n*_CTRL[IOSEL] = 0. Select input 0. • Set EV*n*_CTRL[IOCOND] = 0x3. Input 0 is high. • Set EV*n*_CTRL[MATCHSEL] = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Write 0 to EV*n*_CTRL[STATEV] and 1 to EV*n*_CTRL[STATELD]. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.

Table continues on the next page...

Table 434. SCT configuration example: settings (continued)

Configuration	Register	Setting
Configure states to enable event 0	OUT0_STATE	Set STATEMSK0 bit 0 to 1, enabling event 0 in state 0. Set all other bits to 0.
Configure states to enable event 1	EV1_STATE	Set STATEMSK1 bit 0 to 1, enabling event 1 in state 0. Set all other bits to 0.
Configure states to enable event 2	EV2_STATE	Set STATEMSK2 bit 0 to 1, enabling event 2 in state 0. Set all other bits to 0.
Configure states to enable event 3	EV3_STATE	Set STATEMSK3 bit 1 to 1, enabling event 3 in state 1. Set all other bits to 0.
Configure states to enable event 4	EV4_STATE	Set STATEMSK4 bit 1 to 1, enabling event 4 in state 1. Set all other bits to 0.
Configure states to enable event 5	EV5_STATE	Set STATEMSK5 bit 1 to 1, enabling event 5 in state 1. Set all other bits to 0.

49.7 Memory map and register definition

49.7.1 Register configuration

For most of the SCT registers, the register function depends on the settings of certain other register fields:

- CONFIG[UNIFY] determines whether SCT is used as one 32-bit register (for operation as one 32-bit counter or timer) or as two 16-bit counters or timers named L and H.
  - If CONFIG[UNIFY] = 1, only one register is used (for operation as one 32-bit counter or timer).
  - If CONFIG[UNIFY] = 0, you can access the L and H registers using a 32-bit read or write operation, or you can read or write to them individually.

Typically, you configure CONFIG[UNIFY] by writing to SCT Configuration (CONFIG) before accessing any other registers.

- REGMOD[REGMOD_H*n*] and REGMOD[REGMOD_L*n*] determine whether each set of match and capture registers uses the match or capture functionality:
  - If REGMODE*n* = 0, registers operate as match and reload registers.
  - If REGMODE*n* = 1, registers operate as capture and capture control registers.

49.7.2 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. The following figure shows the registers and register fields that you must configure for each event. In Figure 259, the following letters are used to represent the maximum quantity of certain SCT features:

- i = match and captures
- j = events
- k = states
- p = outputs
- q = inputs

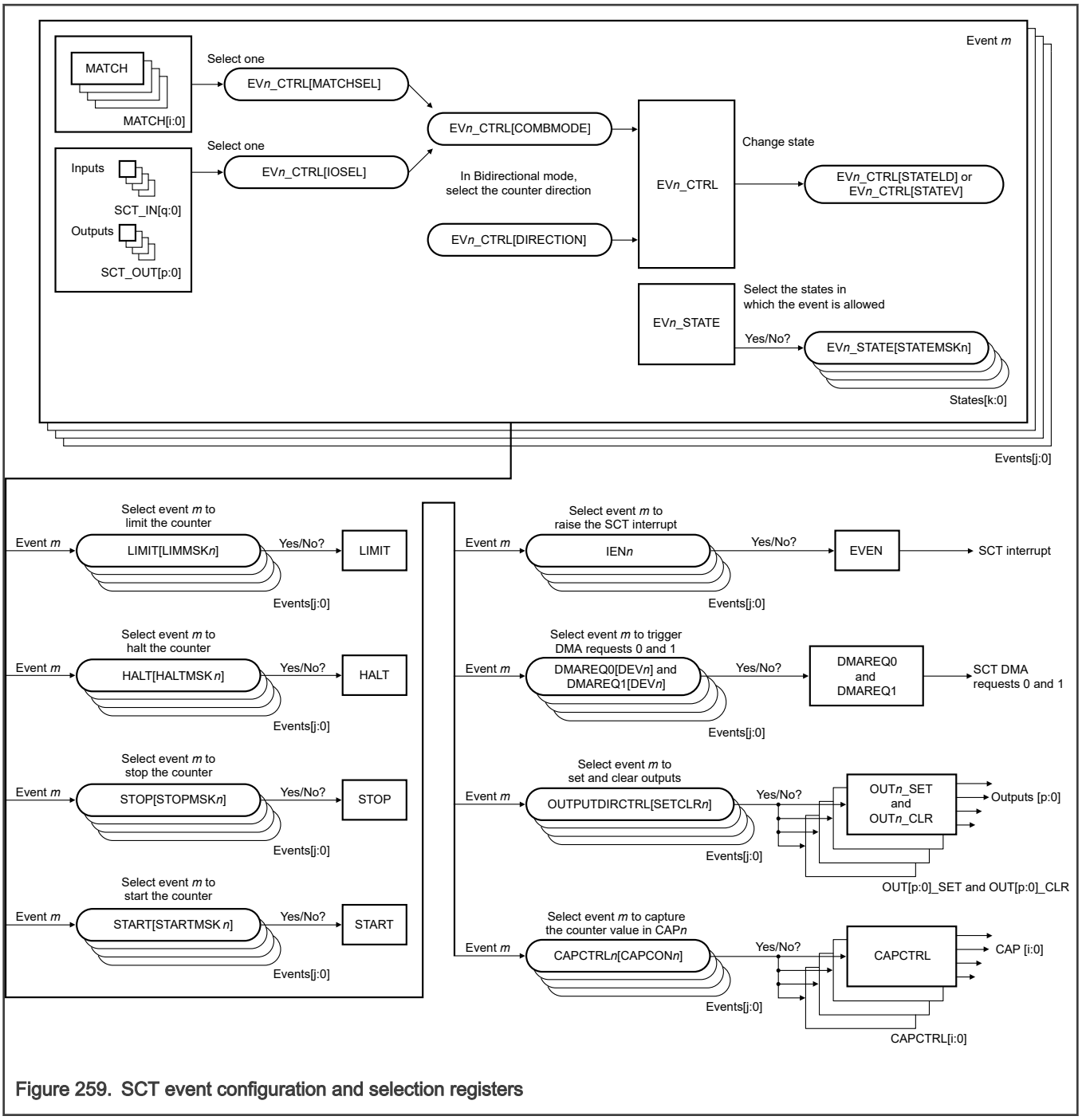


Figure 259. SCT event configuration and selection registers

49.7.2.1 Counter configuration and control registers

SCT contains these registers for configuration, monitoring, and control:

- SCT Configuration (CONFIG) configures SCT in a single, 32-bit counter mode or in dual, 16-bit counter mode. This register helps you perform clocking, clock synchronization, configure automatic limits, and use reload registers.
- SCT Control (CTRL) helps you monitor and set the counter direction. Additionally, using this register, you can clear, start, stop, or halt the 32-bit counter or individual 16-bit counters in Dual-Counter mode.

49.7.2.2 Event configuration registers

Each event is associated with these registers:

- Event n State (EV0_STATE - EV15_STATE) to enable the event
- Event n Control (EV0_CTRL - EV15_CTRL) to define what triggers the event

49.7.2.3 Match and capture registers

SCT includes a set of registers to store the SCT match or capture values:

- Each Match Value (MATCH0 - MATCH15) is associated with a corresponding Match Reload Value (MATCHREL0 - MATCHREL15), which automatically reloads Match Value (MATCH0 - MATCH15) at the beginning of each counter cycle.
- One Match and Capture Register Mode (REGMODE) to configure each match and capture register for either storing a match value or capture value.
- A set of match and capture registers with each register, depending on the setting of Match and Capture Register Mode (REGMODE), for either storing a match value or counter capture value.
- One Capture Control (CAPCTRL0 - CAPCTRL15) for each Capture Value (CAP0 - CAP15).

Figure 260 shows the match logic for SCT.

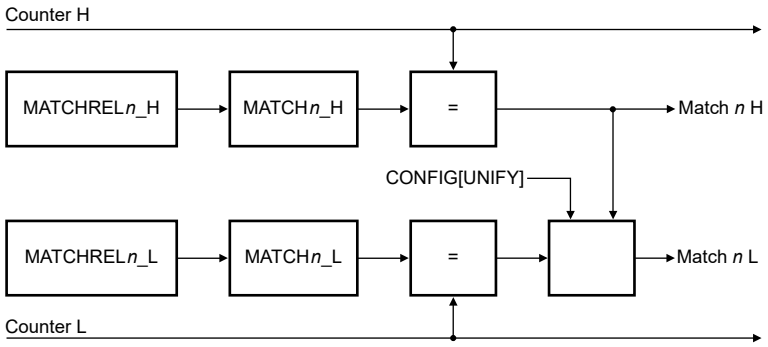


Figure 260. Match logic

Figure 261 shows the capture logic for SCT.

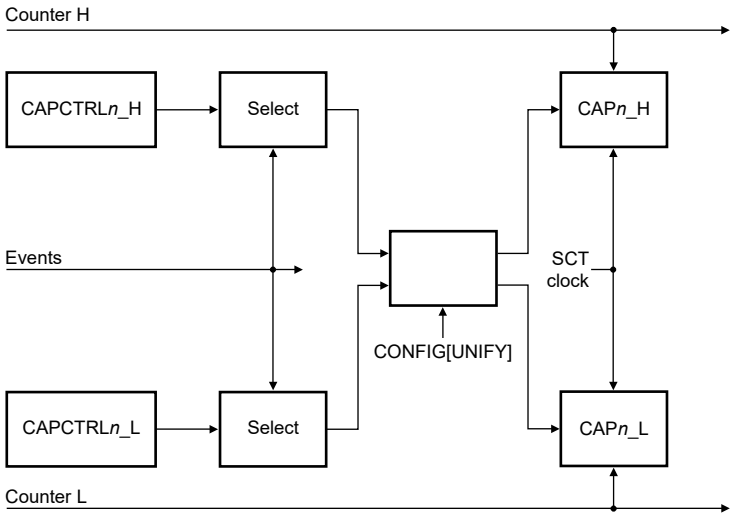


Figure 261. Capture logic

49.7.2.4 Event select registers for counter operations

This group of registers helps you select the events that affect the counter. The counter actions are limit, halt, start, and stop, which apply to the unified counter or to the two 16-bit counters:

- SCT Limit Event Select (LIMIT) selects the events that limit the counter.
- Start Event Select (START) and Stop Event Select (STOP) select the events that start or stop the counter.
- Halt Event Select (HALT) selects the events that halt the counter.
- Counter Value (COUNT) contains the counter value (or values) in the dual-counter setup.

All the aforementioned registers except Start Event Select (START) contain one field per event that selects whether each event limits, stops, starts, or halts the counter (or counters) in Dual-Counter mode. In this mode, you can select the events independently for each counter.

49.7.2.5 Event select registers for setting or clearing outputs

This group of registers helps you select the events that affect the level of each SCT output. Also included in this group are registers to manage conflicts that occur when events try to set or clear the same output:

- One Output n Set (OUT0_SET - OUT9_SET) for each output to select the events that set the output
- One Output n Clear (OUT0_CLR - OUT9_CLR) for each output to select the events that clear the output
- Output Conflict Resolution (RES) that defines an action when multiple events try to control an output at the same time
- Conflict Flag (CONFLAG) and Conflict Interrupt Enable (CONEN) that monitor interrupts arising from "output set and clear conflicts"
- Output Counter Direction Control (OUTPUTDIRCTRL) that interchanges the set and clear output operation caused by an event in Bidirectional mode

Output n Set (OUT0_SET - OUT9_SET) and Output n Clear (OUT0_CLR - OUT9_CLR) contain one field each per event that selects whether the event changes the state for a given output *n*.

In Dual-Counter mode, you can select the events independently for each output.

The following figure shows an SCT output slice.

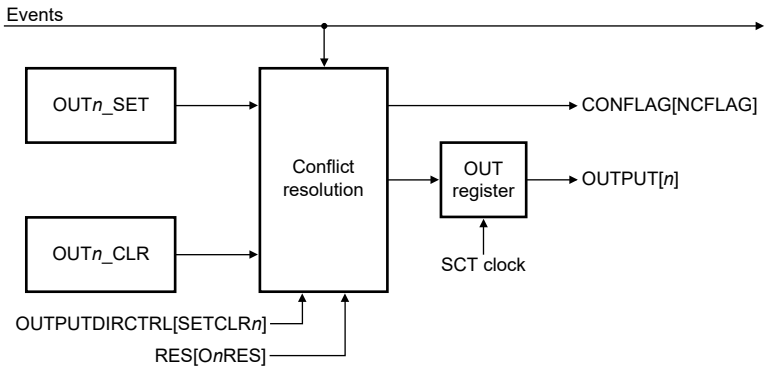


Figure 262. Output slice *n*

49.7.2.6 Event select registers for capturing a counter value

This group of registers helps you select events that capture the counter value and store it in one of Capture Value (CAP0 - CAP15). Each capture register *n* includes an associated Capture Control (CAPCTRL0 - CAPCTRL15), which in turn selects the events to capture the counter value.

49.7.2.7 Event select register for initiating DMA transfers

This register allows each of the two DMA requests to select the events that can generate a DMA request.

DMA Request 0 (DMAREQ0) and DMA Request 1 (DMAREQ1) contain a field for each event that selects whether the event generates a DMA request. An additional field enables the DMA trigger after the match registers are reloaded.

49.7.2.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that request an interrupt:

- Event Flag (EVFLAG) provides a flag for each event that is set when the event occurs.
- Event Interrupt Enable (EVEN) provides a field for each event to be enabled for the SCT interrupt.

This following figure shows an event triggering SCT to generate an interrupt.

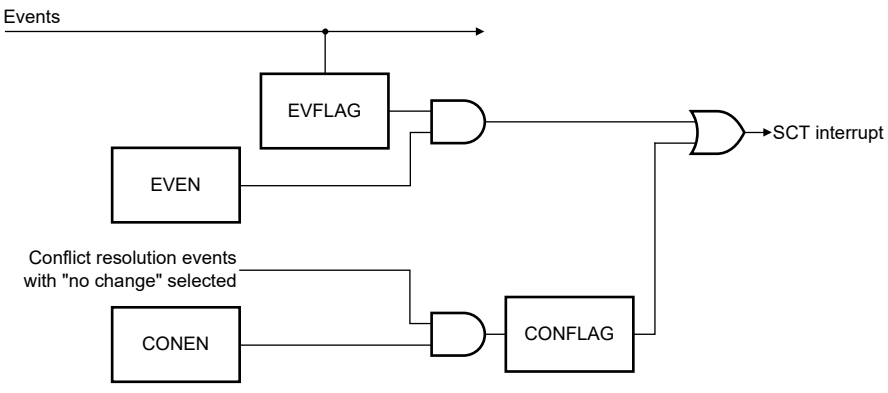


Figure 263. SCT interrupt generation

49.7.2.9 Registers for controlling SCT inputs and outputs

The following registers allow you (as opposed to events) to set SCT input and outputs:

- Input State (INPUT) to read the state of any of the SCT inputs
- Output State (OUTPUT) to set or clear any of the SCT outputs or to read their states

49.7.3 SCT register descriptions

49.7.3.1 SCT memory map

SCT0 base address: 4009_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SCT Configuration (CONFIG)	32	RW	See section
4h	SCT Control (CTRL)	32	RW	See section
8h	SCT Limit Event Select (LIMIT)	32	RW	0000_0000h
Ch	Halt Event Select (HALT)	32	RW	0000_0000h
10h	Stop Event Select (STOP)	32	RW	0000_0000h
14h	Start Event Select (START)	32	RW	0000_0000h
18h	Dither Condition (DITHER)	32	RW	0000_0000h
40h	Counter Value (COUNT)	32	RW	0000_0000h
44h	State Variable (STATE)	32	RW	See section
48h	Input State (INPUT)	32	R	See section
4Ch	Match and Capture Register Mode (REGMODE)	32	RW	0000_0000h
50h	Output State (OUTPUT)	32	RW	See section
54h	Output Counter Direction Control (OUTPUTDIRCTRL)	32	RW	0000_0000h
58h	Output Conflict Resolution (RES)	32	RW	0000_0000h
5Ch	DMA Request 0 (DMAREQ0)	32	RW	See section
60h	DMA Request 1 (DMAREQ1)	32	RW	See section
F0h	Event Interrupt Enable (EVEN)	32	RW	See section
F4h	Event Flag (EVFLAG)	32	RW	See section
F8h	Conflict Interrupt Enable (CONEN)	32	RW	See section
FCh	Conflict Flag (CONFLAG)	32	RW	See section
100h - 13Ch	Capture Value (CAP0 - CAP15)	32	RW	0000_0000h
100h - 13Ch	Match Value (MATCH0 - MATCH15)	32	RW	0000_0000h
140h - 154h	Fractional Match (FRACMAT0 - FRACMAT5)	32	RW	0000_0000h
200h - 23Ch	Capture Control (CAPCTRL0 - CAPCTRL15)	32	RW	0000_0000h
200h - 23Ch	Match Reload Value (MATCHREL0 - MATCHREL15)	32	RW	0000_0000h
240h - 254h	Fractional Match Reload (FRACMATREL0 - FRACMATREL5)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
300h	Event n State (EV0_STATE)	32	RW	See section
304h	Event n Control (EV0_CTRL)	32	RW	See section
308h	Event n State (EV1_STATE)	32	RW	See section
30Ch	Event n Control (EV1_CTRL)	32	RW	See section
310h	Event n State (EV2_STATE)	32	RW	See section
314h	Event n Control (EV2_CTRL)	32	RW	See section
318h	Event n State (EV3_STATE)	32	RW	See section
31Ch	Event n Control (EV3_CTRL)	32	RW	See section
320h	Event n State (EV4_STATE)	32	RW	See section
324h	Event n Control (EV4_CTRL)	32	RW	See section
328h	Event n State (EV5_STATE)	32	RW	See section
32Ch	Event n Control (EV5_CTRL)	32	RW	See section
330h	Event n State (EV6_STATE)	32	RW	See section
334h	Event n Control (EV6_CTRL)	32	RW	See section
338h	Event n State (EV7_STATE)	32	RW	See section
33Ch	Event n Control (EV7_CTRL)	32	RW	See section
340h	Event n State (EV8_STATE)	32	RW	See section
344h	Event n Control (EV8_CTRL)	32	RW	See section
348h	Event n State (EV9_STATE)	32	RW	See section
34Ch	Event n Control (EV9_CTRL)	32	RW	See section
350h	Event n State (EV10_STATE)	32	RW	See section
354h	Event n Control (EV10_CTRL)	32	RW	See section
358h	Event n State (EV11_STATE)	32	RW	See section
35Ch	Event n Control (EV11_CTRL)	32	RW	See section
360h	Event n State (EV12_STATE)	32	RW	See section
364h	Event n Control (EV12_CTRL)	32	RW	See section
368h	Event n State (EV13_STATE)	32	RW	See section
36Ch	Event n Control (EV13_CTRL)	32	RW	See section
370h	Event n State (EV14_STATE)	32	RW	See section
374h	Event n Control (EV14_CTRL)	32	RW	See section
378h	Event n State (EV15_STATE)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
37Ch	Event n Control (EV15_CTRL)	32	RW	See section
500h	Output n Set (OUT0_SET)	32	RW	See section
504h	Output n Clear (OUT0_CLR)	32	RW	See section
508h	Output n Set (OUT1_SET)	32	RW	See section
50Ch	Output n Clear (OUT1_CLR)	32	RW	See section
510h	Output n Set (OUT2_SET)	32	RW	See section
514h	Output n Clear (OUT2_CLR)	32	RW	See section
518h	Output n Set (OUT3_SET)	32	RW	See section
51Ch	Output n Clear (OUT3_CLR)	32	RW	See section
520h	Output n Set (OUT4_SET)	32	RW	See section
524h	Output n Clear (OUT4_CLR)	32	RW	See section
528h	Output n Set (OUT5_SET)	32	RW	See section
52Ch	Output n Clear (OUT5_CLR)	32	RW	See section
530h	Output n Set (OUT6_SET)	32	RW	See section
534h	Output n Clear (OUT6_CLR)	32	RW	See section
538h	Output n Set (OUT7_SET)	32	RW	See section
53Ch	Output n Clear (OUT7_CLR)	32	RW	See section
540h	Output n Set (OUT8_SET)	32	RW	See section
544h	Output n Clear (OUT8_CLR)	32	RW	See section
548h	Output n Set (OUT9_SET)	32	RW	See section
54Ch	Output n Clear (OUT9_CLR)	32	RW	See section

49.7.3.2 SCT Configuration (CONFIG)

Offset

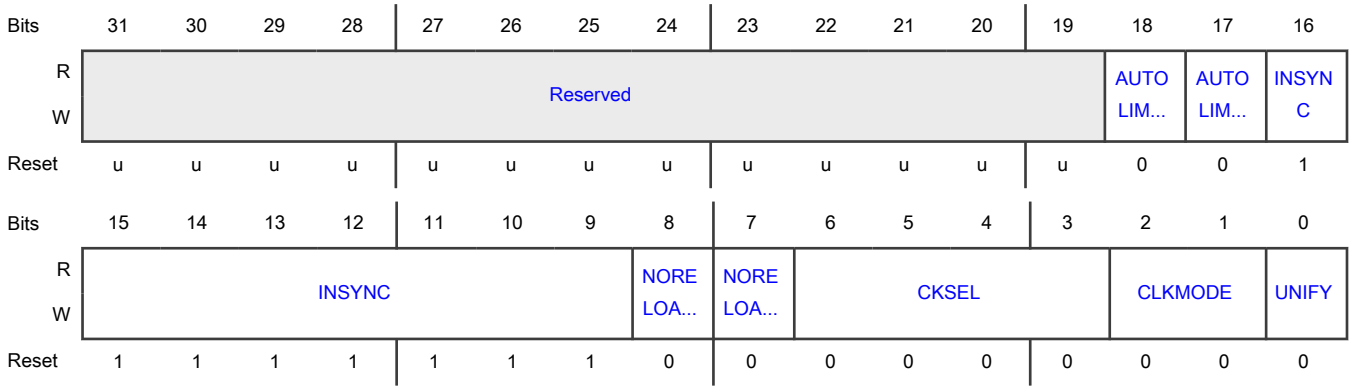
Register	Offset
CONFIG	0h

Function

Configures the overall SCT operation. You must write to this register before writing to any other registers.

Only word writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-19 —	Reserved
18 AUTOLIMIT_H	Auto Limit Higher Enables the counter to be cleared to zero in Unidirectional mode or to change the direction of count in Bidirectional mode, as with any SCT Limit Event Select (LIMIT) event. You can write to this field at any time, except when CONFIG[UNIFY] = 1. If this field = 1, a match on match register 0 is considered as the limit condition without the need to define an associated event. 0b - Disables 1b - Enables
17 AUTOLIMIT_L	Auto Limit Lower Enables the counter to be cleared to zero in Unidirectional mode or to change the direction of count in Bidirectional mode, as with any SCT Limit Event Select (LIMIT) event. You can write to this field at any time, and the field applies to both the higher and lower registers when CONFIG[UNIFY] = 1. If this field = 1, a match on match register 0 is considered as the limit condition. You need not define an associated event. 0b - Disables 1b - Enables
16-9 INSYNC	Input Synchronization Specifies synchronization for input *n* (bit 9 = input 0, bit 10 = input 1, bit 16 = input 7, and so on). If you write 1 to any of these bits, it subjects the corresponding input to synchronization with the SCT clock, before it is used to create an event. If an input is already known to be synchronous with the SCT clock, you may write 0 to this field for a faster input response.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE SCT clock is the system clock for clock modes 0-2. It is the selected, asynchronous SCT input clock for clock mode 3. This field only affects inputs used for event generation, and does not apply to the clock input specified in CONFIG[CKSEL].
8 NORELOAD_H	No Reload Higher Match Specifies whether higher match registers are reloaded from their respective reload registers. Writing 1 to this field eliminates the need to write to Match Reload Value (MATCHREL0 - MATCHREL15) if the match values are fixed. It prevents the higher match registers to be reloaded from their respective reload registers. You can write to this field at any time, except when CONFIG[UNIFY] = 1. 0b - Reloaded 1b - Not reloaded
7 NORELOAD_L	No Reload Lower Match Specifies whether lower match registers are reloaded from their respective reload registers. Writing 1 to this field eliminates the need to write to Match Reload Value (MATCHREL0 - MATCHREL15) if the match values are fixed. It prevents the lower match registers to be reloaded from their respective reload registers. You can write to this field at any time, and the field applies to both the higher and lower registers when CONFIG[UNIFY] = 1. 0b - Reloaded 1b - Not reloaded
6-3 CKSEL	SCT Clock Select Specifies SCT clock. The specific functionality of the designated input or edge is dependent on the mode you select using CONFIG[CLKMODE]. 0000b - Rising edges on input 0 0001b - Falling edges on input 0 0010b - Rising edges on input 1 0011b - Falling edges on input 1 0100b - Rising edges on input 2 0101b - Falling edges on input 2 0110b - Rising edges on input 3 0111b - Falling edges on input 3 1000b - Rising edges on input 4

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1001b - Falling edges on input 4 1010b - Rising edges on input 5 1011b - Falling edges on input 5 1100b - Rising edges on input 6 1101b - Falling edges on input 6 1110b - Rising edges on input 7 1111b - Falling edges on input 7
2-1 CLKMODE	SCT Clock Mode Specifies SCT clock mode. • If this field = 0, SCT uses System Clock mode. In this mode, the system clock clocks the entire SCT module, including all counters and counter prescalers. • If this field = 1, SCT uses Sampled System Clock mode. In this mode, the system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input that CONFIG[CKSEL] selects. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This is a high-performance mode. • If this field = 10, SCT uses Input Clock mode. The input or edge that CONFIG[CKSEL] selects clocks the SCT module, including all counters and prescalers, after first being synchronized with the system clock. The minimum pulse width on the clock input is one bus clock period. This is a low-power, sampled-clock mode. • If this field = 11, SCT uses Asynchronous mode. In this mode, the input or edge that CONFIG[CKSEL] selects clocks the entire SCT module. The SCT outputs are switched synchronously to the SCT input clock, and not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock. 00b - System Clock mode 01b - Sampled System Clock mode 10b - SCT Input Clock mode 11b - Asynchronous mode
0 UNIFY	SCT Operation Determines whether SCT is used as a single, unified 32-bit counter or timer or as dual, independent 16-bit counters or timers named low (L) and high (H). You must define the setting of this field before accessing other registers. 0b - Dual counters, COUNTER_L and COUNTER_H 1b - Unified counter

49.7.3.3 SCT Control (CTRL)

Offset

Register	Offset
CTRL	4h

Function

Allows SCT control.

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only the _L fields are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, CTRL_L and CTRL_H. You can read or write to both these registers, individually.

You can write to all fields in this register when the counter is stopped or halted. When the counter is running, the only fields that you can write to are STOP or HALT. You can write to the other fields in a subsequent write, after HALT becomes 1.

NOTE

- If you select CLKMODE = 0x3, wait for at least 12 system clock cycles between a write access to the H, L, or the unified version of this register and the next write access. This restriction does not apply when writing to the HALT field(s) and then writing to this register again to restart the counters (for example, you must update Match Value (MATCH0 - MATCH15), and this is allowed only when the counters are halted).
- If SCT is operating as two 16-bit counters, events can only modify the state of the outputs when no counter is halted. This is true regardless of what triggered the event.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				PRE_H							BIDIR_H	CLRCTR_H	HALT_H	STOP_H	DOWN_H
Reset	u	u	u	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				PRE_L							BIDIR_L	CLRCTR_L	HALT_L	STOP_L	DOWN_L
Reset	u	u	u	0	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31-29 —	Reserved
28-21	Prescaler for High Counter

Table continues on the next page...

Table continued from the previous page...

Field	Function
PRE_H	Specifies the factor by which the SCT clock is prescaled to produce the high counter clock, which is clocked at the rate of SCT clock ÷ PREL_H + 1. **NOTE** Clear the counter (by writing 1 to CLRCTR) when changing the value of this field.
20 BIDIR_H	Bidirectional Select High Selects the direction of the high counter. If this field = 0, the high counter counts up to its limit condition, and is then cleared to zero. If this field = 1, the high counter counts up to its limit, and then counts down to a limit condition or to zero. 0b - Up 1b - Up-down
19 CLRCTR_H	Clear Counter High Clears the high counter. Writing 1 to this field clears the high counter. CLRCTR_H always reads as 0.
18 HALT_H	Halt Counter High Halts the high counter. When this field = 1, the H counter does not run and no events can occur. Also, CTRL[STOP_H] clears. You can remove the halt condition when SCT is not running (is in a stop condition), by using a single write to this register, to simultaneously clear the HALT and STOP fields. **NOTE** Once set, HALT_H can only be cleared by software to restore counter operation. HALT_H = 1 on reset. 0b - Disable 1b - Enable
17 STOP_H	Stop Counter High Stops the high counter from running. When STOP_H = 1 and HALT = 0, the H counter does not run but I/O events related to the counter can occur. If such an event matches the mask in Start Event Select (START), CTRL[STOP_H] is cleared and counting resumes. 0b - Disabled 1b - Enabled
16 DOWN_H	Down Counter High Specifies whether the high counter is counting up or down.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Hardware sets DOWN_H when the counter is counting up, a counter limit condition occurs, and BIDIR = 1. Hardware clears DOWN_H when the counter is counting down and a limit condition occurs, or when the counter reaches 0. 0b - Up 1b - Down
15-13 —	Reserved
12-5 PRE_L	Prescaler for Low Counter Specifies the factor by which the SCT clock is prescaled to produce the low or unified counter clock. The counter clock is clocked at the rate of SCT clock ÷ PRE_L + 1. **NOTE** You can clear the counter by writing 1 to CLRCTR when changing the value of this field.
4 BIDIR_L	Bidirectional Select Low Selects the direction of the low or unified counter. If this field = 0, the counter counts up to a limit condition and is then cleared to 0. If this field = 1, the counter counts up to a limit, and then counts down to a limit condition or to 0. 0b - Up 1b - Up-down
3 CLRCTR_L	Clear Counter Low Clears the low or unified counter. Writing 1 to this field clears the low or unified counter. The field always reads 0.
2 HALT_L	Halt Counter Low Specifies whether the L or unified counter is enabled. If this field = 1, the low or unified counter does not run and no events can occur. STOP_L also becomes 0. It is possible to remove the halt condition while SCT is in the stop condition (not running), with a single write to this register, to simultaneously clear the HALT field and set the STOP field. Once set, only software can clear HALT_L to restore counter operation. HALT_L = 1 on reset. 0b - Disabled 1b - Enabled
1 STOP_L	Stop Counter Low Specifies whether the low or unified counter is enabled. If STOP_L = 1 and HALT = 0, the low or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, STOP_L is cleared and counting resumes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled 1b - Enabled
0 DOWN_L	Down Counter Low Specifies whether the low or unified counter is counting up or down. Hardware sets DOWN_L when the counter is counting up, counter limit occurs, and BIDIR = 1. Hardware clears this field when the counter is counting down and a limit condition occurs, or when the counter reaches 0. 0b - Up 1b - Down

49.7.3.4 SCT Limit Event Select (LIMIT)

Offset

Register	Offset
LIMIT	8h

Function

Selects the SCT limit event.

An event can limit a running counter. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes the counting direction, if it is in Bidirectional mode. Each field of this register is associated with a different event (bit 0 with event 0, and so on). Writing 1 to a field causes its associated event to serve as a LIMIT event, which whenever occurs, resets the counter to zero in Unidirectional mode, or changes the count direction in Bidirectional mode and keeps running. To define the actual limiting event (a match, an I/O pin toggle, and so on), see Event n Control (EV0_CTRL - EV15_CTRL).

NOTE

Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

In addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. CONFIG[AUTOLIMIT_L] and CONFIG[AUTOLIMIT_H] enable or disable this feature.

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only the _L fields are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, LIMIT_L and LIMIT_H. You can read or write to both the _L and _H registers individually.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LIMMSK_H															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LIMMSK_L															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 LIMMSK_H	Limit Event Counter High Specifies the counter limit for the high counter. If bit *n* is 1, event *n* is used as a counter limit for the high counter (event 0 = bit 16, event 1 = bit 17, and so on). Number of bits = number of events in this SCT.
15-0 LIMMSK_L	Limit Event Counter Low Specifies the counter limit for the low counter. If bit *n* is 1, event *n* is used as a counter limit for the low or unified counter (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT.

49.7.3.5 Halt Event Select (HALT)

Offset

Register	Offset
HALT	Ch

Function

Halts high and low events.

A halt event can disable (or halt) a running counter. When any of the events selected in this register occur, the counter stops running and all further events are disabled.

Each field of this register is associated with a different event (bit 0 with event 0, and so on). Setting a bit causes the associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, and so on), see Event n Control (EV0_CTRL - EV15_CTRL).

NOTE

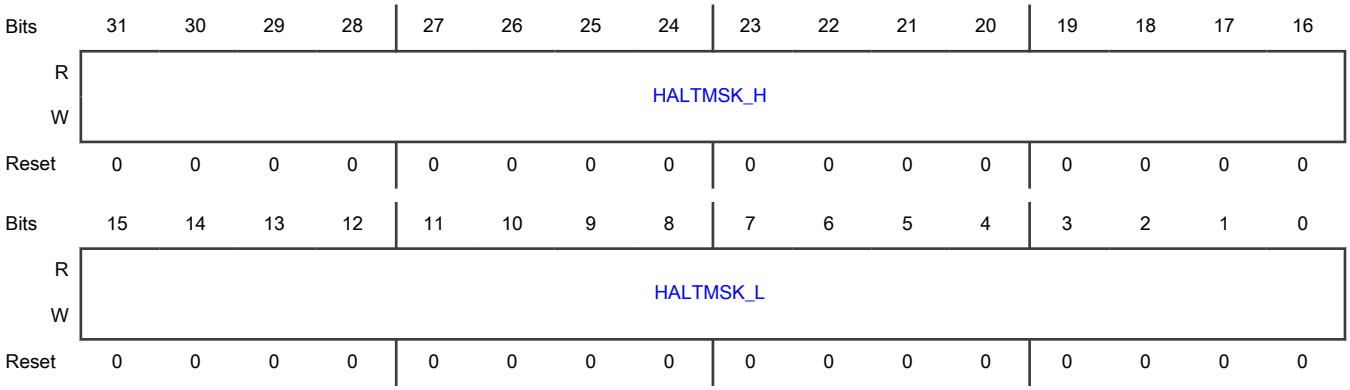
You can remove a HALT condition by clearing CTRL[HALT].

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only the _L fields are used.

- If CONFIG[UNIFY] = 0, you can write to this register as two registers, HALT_L and HALT_H. You can read or write to both the _L and _H registers individually.

Diagram



Fields

Field	Function
31-16 HALTMSK_H	Halt Event High Halts the high event. If bit *n* is 1, event *n* writes 1 to CTRL[HALT_H] (event 0 = bit 16, event 1 = bit 17, and so on). Number of bits = number of events in this SCT.
15-0 HALTMSK_L	Halt Event Low Halts the low event. If bit *n* is 1, event *n* writes 1 to CTRL[HALT_L] (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT.

49.7.3.6 Stop Event Select (STOP)

Offset

Register	Offset
STOP	10h

Function

Stops high and low events.

A stop event can stop a running counter. When any of the events selected in this register occur, counting is suspended. This means that the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in Start Event Select (START), such as an I/O event or an event generated by the other counter, can restart the counter.

This register specifies the events that stop the counter. Each field of the register is associated with a different event (bit 0 with event 0, and so on). Setting a bit causes its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, and so on), see Event n Control (EV0_CTRL - EV15_CTRL).

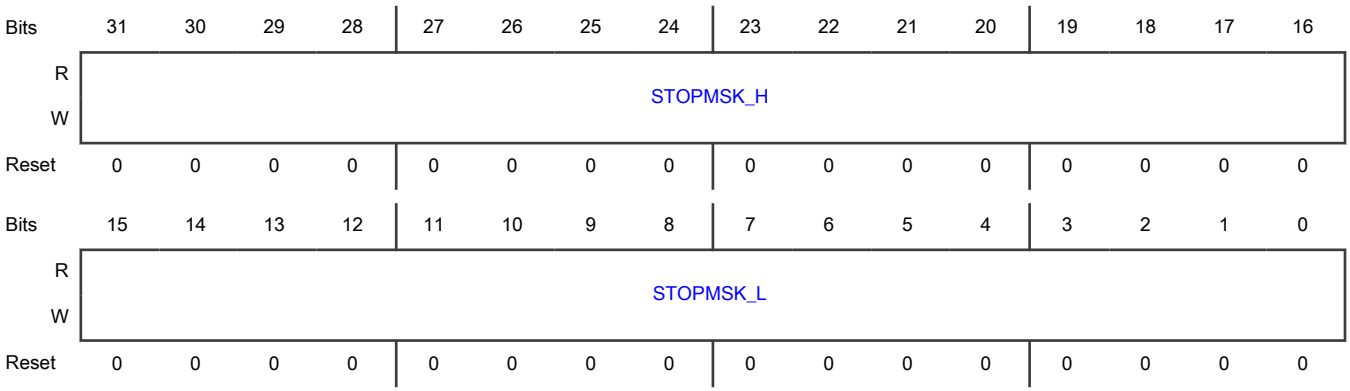
NOTE

You can stop and restart the counter by writing to SCT Control (CTRL).

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only the _L fields are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, STOP_L and STOP_H. You can read or write to both the _L and _H registers individually.

Diagram



Fields

Field	Function
31-16 STOPMSK_H	Stop Event High Stops the high event. If bit *n* is 1, the event *n* writes 1 to CTRL[STOP_H] (event 0 = bit 16, event 1 = bit 17, and so on). Number of bits = number of events in this SCT.
15-0 STOPMSK_L	Stop Event Low Stops the low event. If bit *n* is 1, the event *n* writes 1 to CTRL[STOP_L] (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT.

49.7.3.7 Start Event Select (START)

Offset

Register	Offset
START	14h

Function

Starts high and low events.

A start event can restart a stopped counter.

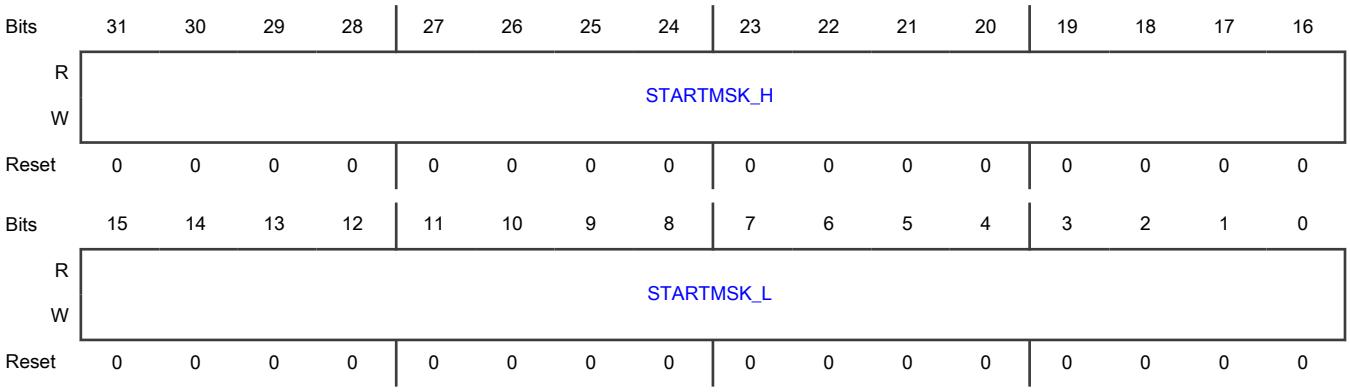
When any of the events selected in Start Event Select (START) occur, counting restarts from the current counter value.

Each field of this register associates with a different event (bit 0 with event 0, and so on). Setting a bit causes the associated event to serve as a START event. When a START event occurs, hardware clears CTRL[STOP]. A START event has no effect on CTRL[HALT]. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in Dual-Counter mode), see Event n Control (EV0_CTRL - EV15_CTRL).

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only the _L fields are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, START_L and START_H. You can read or write to both the _L and _H registers individually.

Diagram



Fields

Field	Function
31-16 STARTMSK_H	Start Event High Starts the high event. If bit *n* is 1, event *n* writes 0 to CTRL[STOP_H] (event 0 = bit 16, event 1 = bit 17, and so on). Number of bits = number of events in this SCT.
15-0 STARTMSK_L	Start Event Low Starts the low event. If bit *n* is 1, event *n* writes 0 to CTRL[STOP_L] (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT.

49.7.3.8 Dither Condition (DITHER)

Offset

Register	Offset
DITHER	18h

Function

Enables the dither engine to advance to the next count in the dither pattern every time the SCT counter reaches zero (that is, at the start of every new SCT counter cycle). Using this register, you can alter this behavior by qualifying the advancement through the dither pattern with designated events. As with the other condition or mask registers (such as Halt Event Select (HALT), Stop Event Select (STOP), and SCT Limit Event Select (LIMIT)), each field of this register corresponds to an event.

Writing 1 to one or more fields in this register causes the dither engine to advance to the next element in the dither pattern (that is, increment the 16-state cycle counter) by only following SCT counter cycles during which one or more of the designated dither events have occurred.

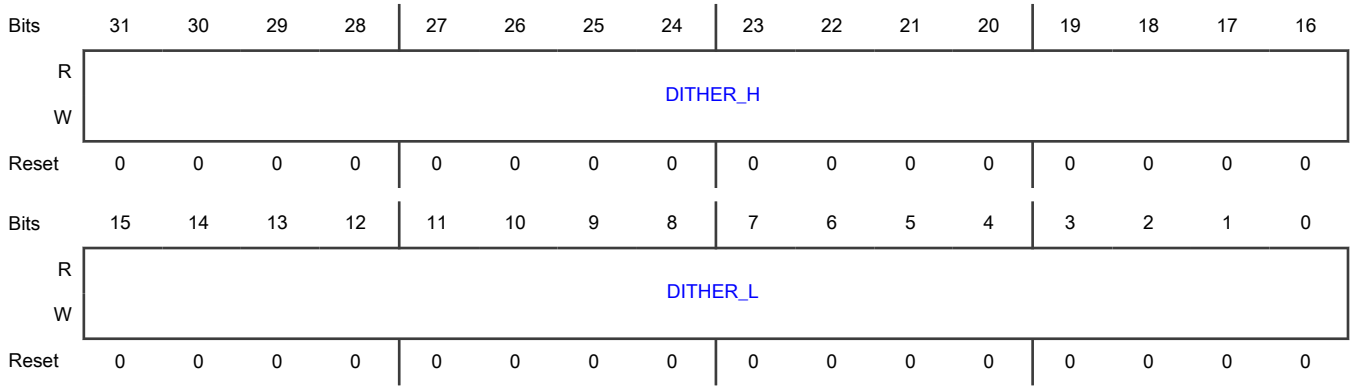
There exists a global dither condition register per 16-bit SCT. This register controls advancement through the dither patterns for all the match registers associated with that half of SCT.

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONGIG[UNIFY] = 1, only the _L bits are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, START_L and START_H. You can read and write to both the _L and _H registers individually or in a single 32-bit read or write operation.

For details on the dither engine and dither pattern, see Dithering.

Diagram



Fields

Field	Function
31-16 DITHER_H	Dither High Dithers high events. If bit *n* is 1, event *n* writes 0 to CTRL[STOP_H] (event 0 = bit 16, event 1 = bit 17, and so on). Number of bits = number of events in this SCT.
15-0 DITHER_L	Dither Low Dithers low events. If bit *n* is 1, event *n* writes 0 to CTRL[STOP_L] (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT.

49.7.3.9 Counter Value (COUNT)

Offset

Register	Offset
COUNT	40h

Function

Defines high and low counters.

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, this register acts as a unified 32-bit register and both the _L and _H fields are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, COUNT_L and COUNT_H. You can read and write to both the _L and _H registers individually. In this case, the _L and _H registers count independently under the control of other registers.

You are allowed to write to COUNT_L, COUNT_H, or the unified register only when the corresponding counter is halted (CTRL[HALT] = 1). In case you attempt to write to the counter when it is not halted, a bus error takes place. However, you can read the counter registers at any time.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTR_H															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTR_L															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 CTR_H	Counter High Allows reads and writes to the upper bits of the unified counter. If CONFIG[UNIFY] = 0, read or write the 16-bit H counter value, and if CONFIG[UNIFY] = 1, read or write the upper 16 bits of the 32-bit unified counter.
15-0 CTR_L	Counter Low Allows reads and writes to the lower bits of the unified counter. If CONFIG[UNIFY] = 0, read or write the 16-bit L counter value, and if CONFIG[UNIFY] = 1, read or write the lower 16 bits of the 32-bit unified counter.

49.7.3.10 State Variable (STATE)

Offset

Register	Offset
STATE	44h

Function

Specifies high and low state variables.

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

NOTE

Event n State (EV0_STATE - EV15_STATE) defines the event that is enabled in each group.

You can read the state associated with a counter at any time, and are allowed to write to STATE[STATE_L], STATE[STATE_H], or the unified register only when the corresponding counter is halted (CTRL[HALT_L] and CTRL[HALT_H] = 1).

The state variable is the main feature that distinguishes SCT from other counters, timers, and PWM blocks. Events can be made to occur only in certain states, and these events can, in turn, perform the following actions:

- Set and clear outputs
- Limit, stop, and start the counter
- Cause interrupts and DMA requests
- Modify the state variable

The application completely controls the value of a state variable. If an application does not use states, the value of the state variable remains zero, which is the default value.

You can use a state variable to track and control multiple cycles of the associated counter in any desired operational sequence.

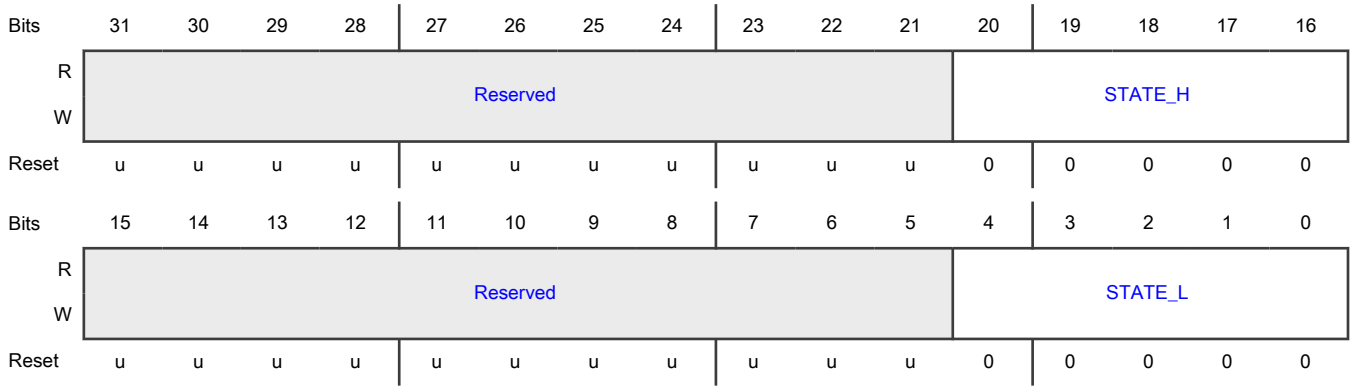
EV*n*_CTRL[STATEV] and EV*n*_CTRL[STATELD] of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only the _L bits are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, STATE_L and STATE_H.

You can read or write to both the _L and _H registers individually.

Diagram



Fields

Field	Function
31-21 —	Reserved
20-16 STATE_H	State Variable High
15-5 —	Reserved
4-0 STATE_L	State Variable Low

49.7.3.11 Input State (INPUT)

Offset

Register	Offset
INPUT	48h

Function

Indicates the state and form of SCT inputs.

You can read the state of SCT inputs, in slightly different forms, using this register:

- The AIN fields of this register indicate the state of the inputs captured on the last rising edge of the SCT clock. This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
- The SIN fields of this register indicate the form of the inputs used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in CONFIG[INSYNC]:
  - If CONFIG[INSYNC] = 1, the input is triple-synchronized to the SCT clock, resulting in a stable signal that is delayed by three SCT clock cycles.

— If CONFIG[INSYNC] = 0, the value of SIN is identical to the value of AIN.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SIN15	SIN14	SIN13	SIN12	SIN11	SIN10	SIN9	SIN8	SIN7	SIN6	SIN5	SIN4	SIN3	SIN2	SIN1	SIN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AIN15	AIN14	AIN13	AIN12	AIN11	AIN10	AIN9	AIN8	AIN7	AIN6	AIN5	AIN4	AIN3	AIN2	AIN1	AIN0
W																
Reset	0	0	0	0	0	0	0	0	u	u	u	u	u	u	u	u

Fields

Field	Function
31 SIN15	Input 15 state
30 SIN14	Input 14 state
29 SIN13	Input 13 state
28 SIN12	Input 12 state
27 SIN11	Input 11 state
26 SIN10	Input 10 state
25 SIN9	Input 9 state
24 SIN8	Input 8 state
23 SIN7	Input 7 state
22	Input 6 state

Table continues on the next page...

Table continued from the previous page...

Field	Function
SIN6	
21 SIN5	Input 5 state
20 SIN4	Input 4 state
19 SIN3	Input 3 state
18 SIN2	Input 2 state
17 SIN1	Input 1 state
16 SIN0	Input 0 state
15 AIN15	Input 15 state
14 AIN14	Input 14 state
13 AIN13	Input 13 state
12 AIN12	Input 12 state
11 AIN11	Input 11 state
10 AIN10	Input 10 state
9 AIN9	Input 9 state
8 AIN8	Input 8 state

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 AIN7	Input 7 state
6 AIN6	Input 6 state
5 AIN5	Input 5 state
4 AIN4	Input 4 state
3 AIN3	Input 3 state
2 AIN2	Input 2 state
1 AIN1	Input 1 state
0 AIN0	Input 0 state

49.7.3.12 Match and Capture Register Mode (REGMODE)

Offset

Register	Offset
REGMODE	4Ch

Function

Specifies whether each register acts as a match register or capture register. SCT contains multiple match and capture registers, and each register includes an accompanying register that functions as a reload register when the primary register is used as a match or capture-control register.

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only REGMODE[REGMOD_L*n*] is used. REGMODE[REGMOD_H*n*] is used only when CONFIG[UNIFY] = 0.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, REGMODE_L and REGMODE_H. You can read or write to both the _L and _H registers individually. REGMODE[REGMOD_L*n*] controls the lower match and capture registers, and REGMODE[REGMOD_H*n*] controls the higher or upper match and capture registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...	REGMOD_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 REGMOD_Hn	Register Mode High Controls match and capture register *n*. If this field = 0, register *n* operates as a match register, and if this field = 1, register *n* operates as a capture register. 0b - Match 1b - Capture
15-0 REGMOD_Ln	Register Mode Low Controls match and capture register *n*. If this field = 0, register *n* operates as a match register, and if this field = 1, register *n* operates as a capture register. 0b - Match 1b - Capture

49.7.3.13 Output State (OUTPUT)

Offset

Register	Offset
OUTPUT	50h

Function

Determines SCT output state.

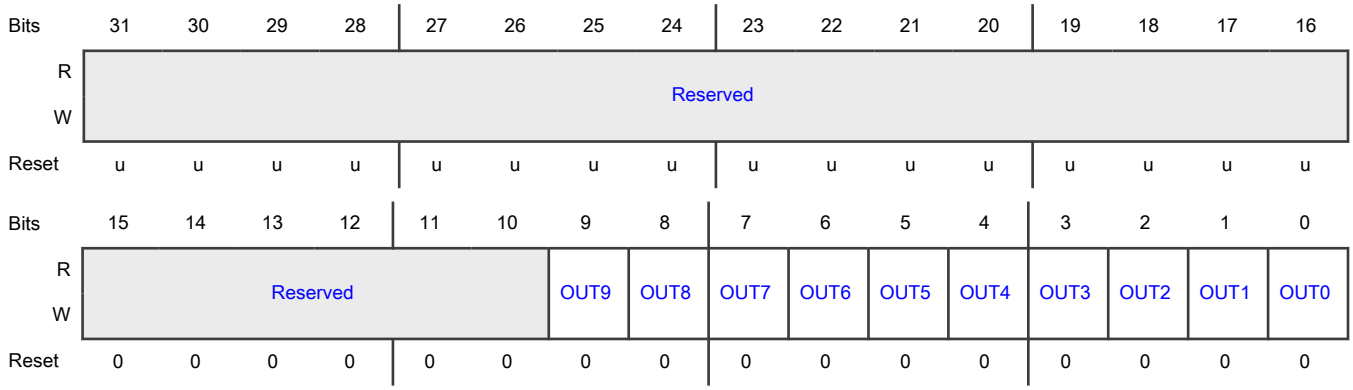
Each SCT output has a corresponding field in this register to allow you to control the output state directly or read its current state.

While the counter is running, only events can set, clear, and toggle these outputs. However, using this register, you can write to any of the output registers when both the counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (CTRL[HALT] = 1).

You can read this register at any time to sense the state of the outputs.

Only word writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-10 —	Reserved
9-0 OUTn	Output Low and High Specifies the SCT output state. The number of bits equals the number of outputs in this SCT. 0b - Forces the corresponding output low 1b - Forces the corresponding output high

49.7.3.14 Output Counter Direction Control (OUTPUTDIRCTRL)

Offset

Register	Offset
OUTPUTDIRCTRL	54h

Function

Specifies, for each output, the impact of the counting direction on the meaning of set and clear operations on the output, when using Bidirectional mode. The purpose of this register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Only word writes are permitted to this register; attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												SETCLR9		SETCLR8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SETCLR7		SETCLR6		SETCLR5		SETCLR4		SETCLR3		SETCLR2		SETCLR1		SETCLR0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-18: SETCLR9 17-16: SETCLR8 15-14: SETCLR7 13-12: SETCLR6 11-10: SETCLR5 9-8: SETCLR4 7-6: SETCLR3 5-4: SETCLR2 3-2: SETCLR1 1-0: SETCLR0	Set and Clear Operation on Output Specifies the set and clear operation on output *n* depending on the counter direction. 00b - Not dependent on the direction of any counter 01b - Reversed when counter L or the unified counter is counting down 10b - Reversed when counter H is counting down (do not use this value when CONFIG[UNIFY] = 1) 11b - Reserved (do not program this value)

49.7.3.15 Output Conflict Resolution (RES)

Offset

Register	Offset
RES	58h

Function

Specifies the action to be taken if multiple events (or even the same event) dictate that a given output must be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, write 1 to the fields for that event in both Output n Set (OUT0_SET - OUT9_SET) and Output n Clear (OUT0_CLR - OUT9_CLR), and write 0x3 to RES[O*n*RES].

Only word writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												O9RES		O8RES	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	O7RES		O6RES		O5RES		O4RES		O3RES		O2RES		O1RES		O0RES	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-18: O9RES 17-16: O8RES 15-14: O7RES 13-12: O6RES 11-10: O5RES 9-8: O4RES 7-6: O3RES 5-4: O2RES 3-2: O1RES 1-0: O0RES	Output Resolution Specifies the effect of simultaneous set and clear on output *n*. 00b - No change 01b - Set output (or clear, based on OUTPUTDIRCTRL[SETCLRn]) 10b - Clear output (or set, based on OUTPUTDIRCTRL[SETCLRn]) 11b - Toggle output

49.7.3.16 DMA Request 0 (DMAREQ0)

Offset

Register	Offset
DMAREQ0	5Ch

Function

Enables DMA requests to be generated when a particular event occurs or when counter match registers are loaded from the reload registers. SCT includes two DMA request outputs.

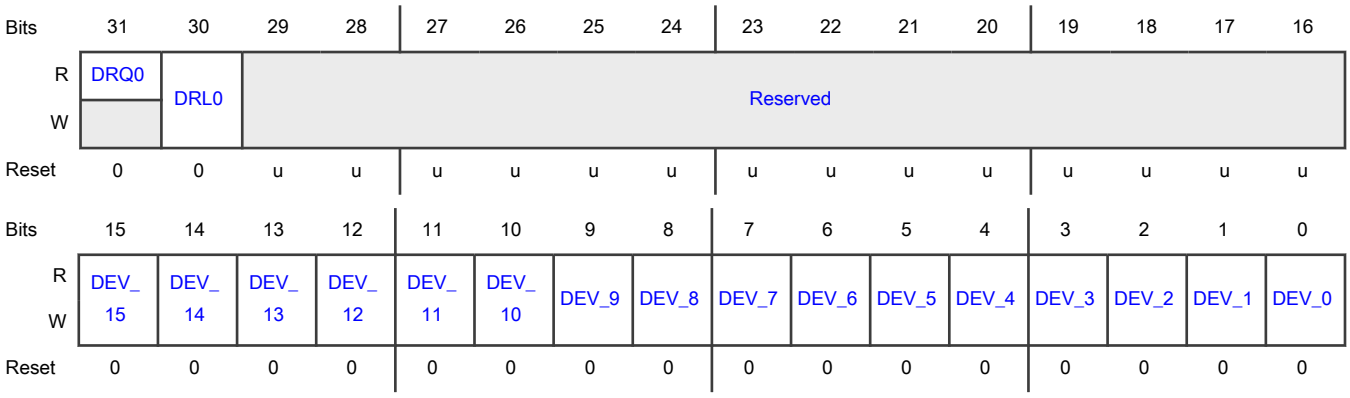
The DMA request registers are word-write only. Attempting to write a halfword value to these registers results in a bus error.

Event-generated DMA requests are particularly useful for launching DMA activity to or from other peripherals under SCT control.

NOTE

In this chip, SCT DMA requests are connected to the DMA trigger inputs, and the DMA trigger input mux registers control them.

Diagram



Fields

Field	Function
31 DRQ0	DMA Request 0 State Indicates the state of DMA request 0. **NOTE** If the related DMA channel is enabled and properly set up, it is unlikely for you to see this flag. The DMA service rapidly clears this field. If the flag remains set, there could be an issue with the DMA setup.
30 DRL0	DMA Request Low 0 Triggers DMA request 0. Writing 1 to this field triggers DMA request 0 when it loads MATCH*n*[MATCH_L] or the unified registers, using MATCHREL*n*[RELOAD_L] or the unified registers.
29-16 —	Reserved
15-0 DEV_n	DMA Request Event Triggers DMA request 0 using event *n*. If bit *n* = 1, event *n* triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT.

49.7.3.17 DMA Request 1 (DMAREQ1)

Offset

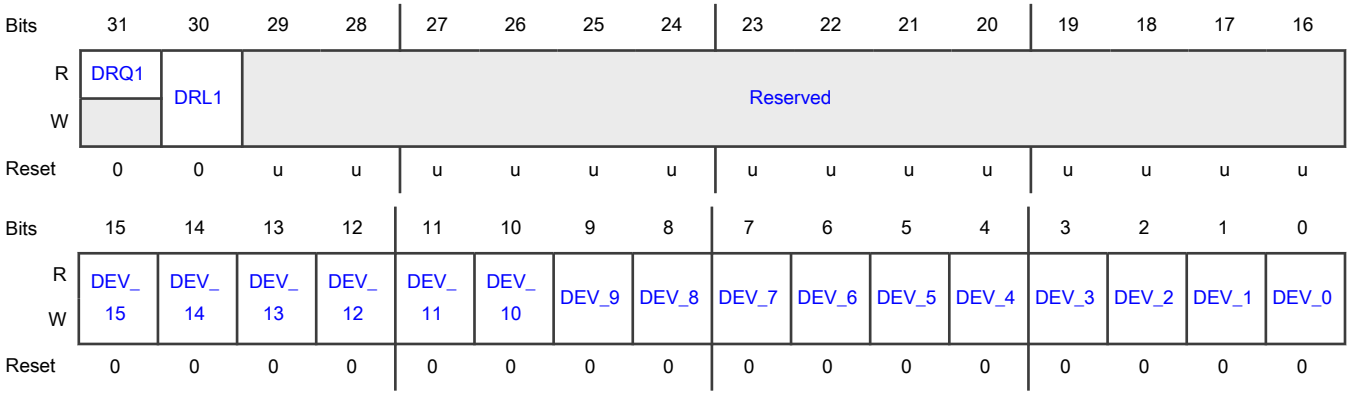
Register	Offset
DMAREQ1	60h

Function

Specifies DMA request 1.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31 DRQ1	DMA Request 1 State Specifies the state of DMA request 1. **NOTE** If the related DMA channel is enabled and properly set up, it is unlikely for you to see this flag, which the DMA service clears rapidly. If the flag remains set, there could be an issue with the DMA setup.
30 DRL1	DMA Request Low 1 Triggers DMA request 1. Writing 1 to this field triggers DMA request 1 when it loads MATCH*n*[MATCH_L] or the unified registers, using MATCHREL*n*[RELOAD_L] or the unified registers.
29-16 —	Reserved
15-0	DMA Request Event

Table continues on the next page...

Table continued from the previous page...

Field	Function
DEV_n	Triggers DMA request using event *n*. If bit *n* = 1, event *n* triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT.

49.7.3.18 Event Interrupt Enable (EVEN)

Offset

Register	Offset
EVEN	F0h

Function

Enables flags to request an interrupt if EVFLAG[FLAG*n*] = 1.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IEN15	IEN14	IEN13	IEN12	IEN11	IEN10	IEN9	IEN8	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-0 IENn	Event Interrupt Enable Enables event interrupts. 0b - Disables 1b - Enables

49.7.3.19 Event Flag (EVFLAG)

Offset

Register	Offset
EVFLAG	F4h

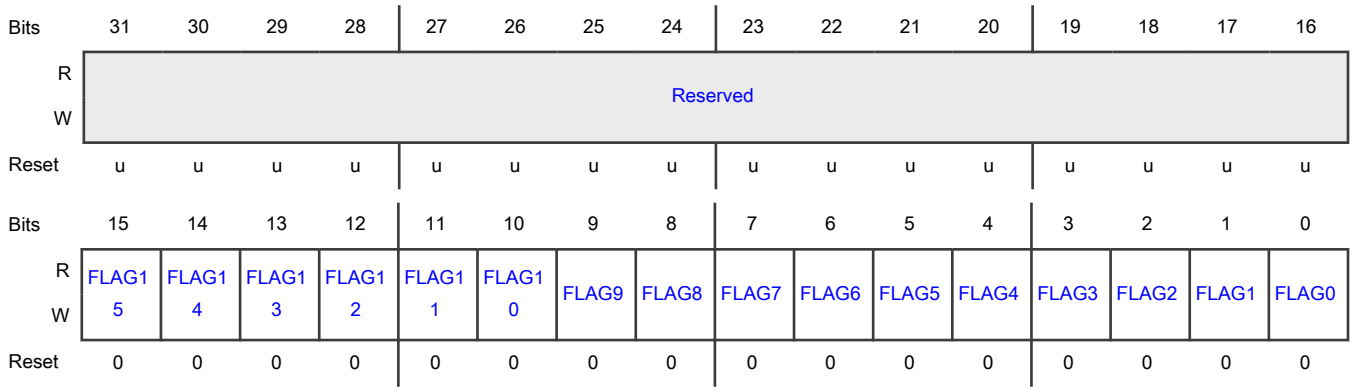
Function

Records events.

Writing 1 to this register clears the corresponding flags and negates the SCT interrupt request if all enabled flags are 0.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 FLAGn	Event Flag Specifies event flag *n*. Bit *n* is 1 if event *n* has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of events in this SCT. 0b - No flag 1b - Event n flag

49.7.3.20 Conflict Interrupt Enable (CONEN)

Offset

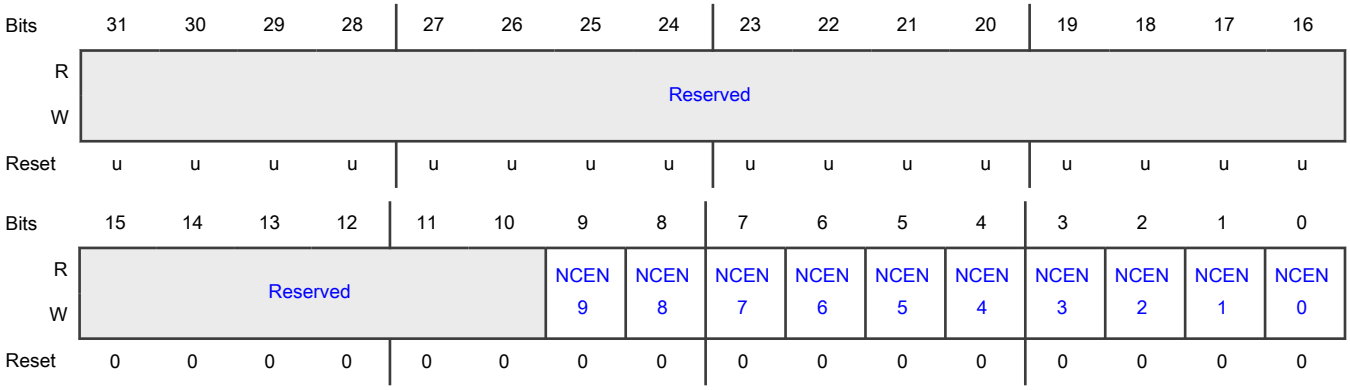
Register	Offset
CONEN	F8h

Function

Enables no-change conflict events specified in Output Conflict Resolution (RES) to generate an interrupt request.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 NCENn	No Change Conflict Event and Interrupt Enable Specifies whether an interrupt is generated. SCT requests an interrupt when bit *n* of this register and Conflict Flag (CONFLAG) are both = 1 (output 0 = bit 0, output 1 = bit 1, and so on). 0b - No interrupt 1b - Interrupt

49.7.3.21 Conflict Flag (CONFLAG)

Offset

Register	Offset
CONFLAG	FCh

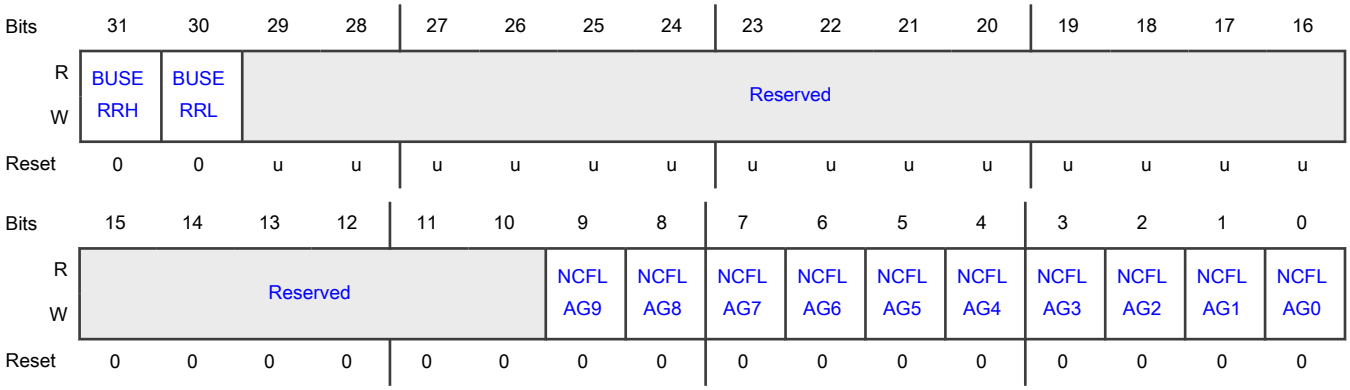
Function

Records a no-change conflict occurrence and also provides details of any bus errors.

Writing 1 to the fields of this register clears the corresponding read bits and negates the SCT interrupt request if all enabled flags are zero.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31 BUSERRH	Bus Error High Specifies whether the bus error is low or unified. The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted.
30 BUSERRL	Bus Error Low or Unified Specifies whether the bus error is low or unified. The most recent bus error from this SCT involved writing to CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful.
29-10 —	Reserved
9-0 NCFLAGn	No Change Conflict Event Flag Specifies whether a no-change conflict event occurred. 0b - Did not occur 1b - Occurred

49.7.3.22 Capture Value (CAP0 - CAP15)

Offset

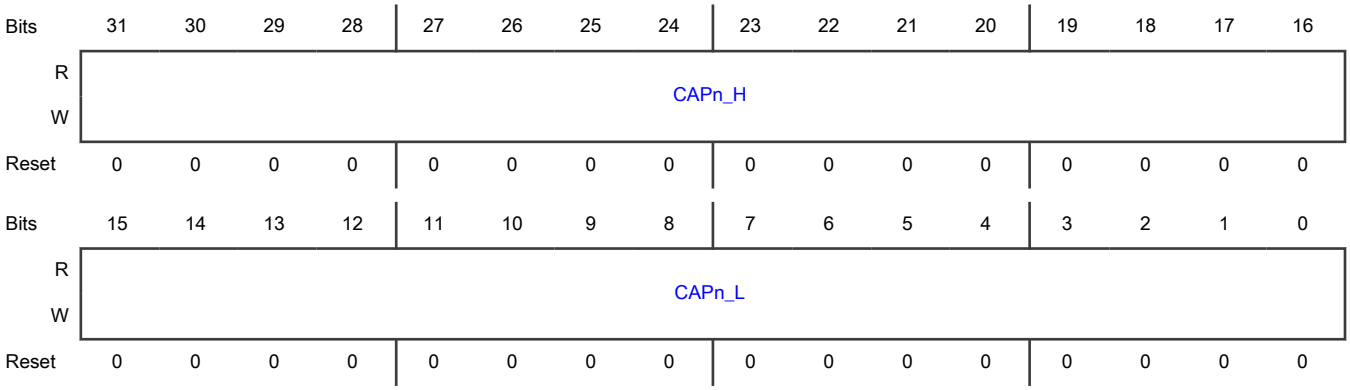
For n = 0 to 15:

Register	Offset
CAPn	100h + (n × 4h)

Function

Allows you to record counter values when the events selected by the corresponding capture control registers occur.

Diagram



Fields

Field	Function
31-16 CAPn_H	Capture High Captures the higher counter value. If CONFIG[UNIFY] = 0, read the 16-bit counter value at which this register was last captured, and if CONFIG[UNIFY] = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.
15-0 CAPn_L	Capture Low Captures the lower counter value. If CONFIG[UNIFY] = 0, read the 16-bit counter value at which this register was last captured, and if CONFIG[UNIFY] = 1, read the lower 16 bits of the 32-bit value where this register was last captured.

49.7.3.23 Match Value (MATCH0 - MATCH15)

Offset

For n = 0 to 15:

Register	Offset
MATCHn	100h + (n × 4h)

Function

Is compared to counters to help create events.

You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 0, the L and H registers are independently compared to the L and H counters.
- If CONFIG[UNIFY] = 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A match can only occur in a clock for which the counter is running (Stop Event Select (STOP) and Halt Event Select (HALT) are 0).

You can read this register at any time. Writing to MATCH[MATCH*n*_L], MATCH[MATCH*n*_H], or the unified register is only allowed when the corresponding counter is halted (CTRL[HALT_L] and CTRL[HALT_H] are 1). Match events occur in the SCT clock for which the counter is (or would be) incremented to the next value. When a match event limits its counter, the value in this register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from reload registers to match registers. Before starting a counter, you can write one value to this register, used in the first cycle of the counter, and a different value to the corresponding match reload register used in the second cycle.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCHn_H															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCHn_L															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 MATCHn_H	Match High Compares upper bits to the unified counter. If CONFIG[UNIFY] = 0, read or write the 16-bit value to be compared to the H counter. If CONFIG[UNIFY] = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.
15-0 MATCHn_L	Match Low Compares lower bits to the unified counter. If CONFIG[UNIFY] = 0, read or write the 16-bit value to be compared to the lower counter. If CONFIG[UNIFY] = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.

49.7.3.24 Fractional Match (FRACMAT0 - FRACMAT5)

Offset

Register	Offset
FRACMAT0	140h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
FRACMAT1	144h
FRACMAT2	148h
FRACMAT3	14Ch
FRACMAT4	150h
FRACMAT5	154h

Function

Provides a higher average resolution over time by applying a dither pattern as described in Dithering. This dither pattern results in delaying recognition of a match for one counter clock for *n* (0 to 15) out of every 16 counter cycles. You must program the value of *n* in these fractional match registers, which are provided for up to first six of the match registers.

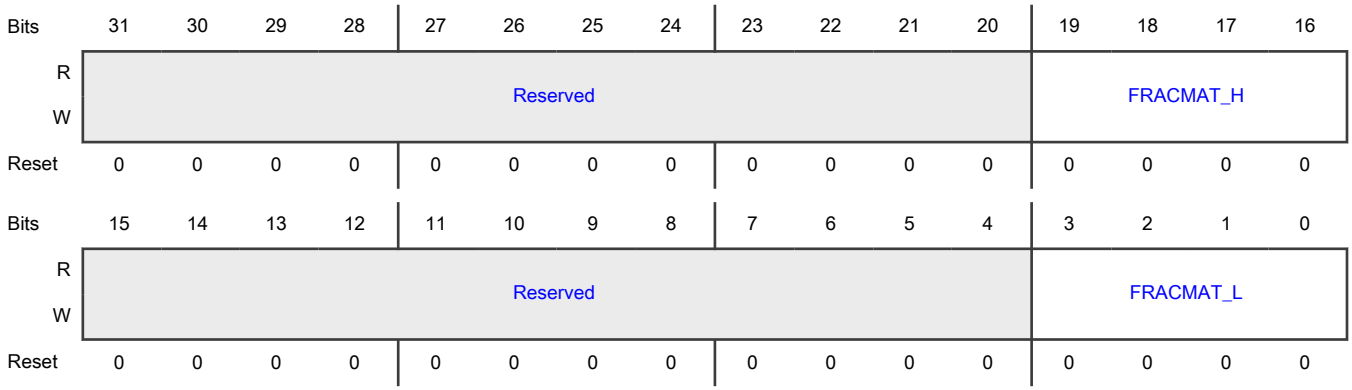
You can read fractional match registers at any time, but can write to FRACMAT*n*[FRACMAT_L], FRACMAT*n*[FRACMAT_H], or the unified register only when the corresponding counter is halted (CTRL[HALT_L] and CTRL[HALT_H] are 1).

Each fractional match register has an associated fractional match reload register. The contents of these reload registers are transferred into the fractional match registers at the start of every new SCT counter cycle unless CONFIG[NORELOAD] = 1 for the appropriate half counter.

You can write to the reload registers at any time, regardless of whether the counter is running.

There is no write-through from the fractional match reload registers to the fractional match registers. Before starting a counter, you can write a value to the fractional match register that is used in the first cycle or period of operation, and a different value to the corresponding fractional match reload register that is used in the second cycle or period.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16	Fractional Match High

Table continues on the next page...

Table continued from the previous page...

Field	Function
FRACMAT_H	Allows reads or writes to upper bits for fractional match. If CONFIG[UNIFY] = 0, read or write a 4-bit value specifying the dither pattern to be applied to the corresponding MATCH*n*[MATCH_H] field (*n* = 0 to 5).
15-4 —	Reserved
3-0 FRACMAT_L	Fractional Match Low Allows reads or writes to lower bits for fractional match. If CONFIG[UNIFY] = 0, read or write a 4-bit value specifying the dither pattern to be applied to the corresponding MATCH*n*[MATCH_L] field (*n* = 0 to 5). If CONFIG[UNIFY] = 1, the value applies to the unified, 32-bit fractional match register.

49.7.3.25 Capture Control (CAPCTRL0 - CAPCTRL15)

Offset

For n = 0 to 15:

Register	Offset
CAPCTRLn	200h + (n × 4h)

Function

Controls the events that cause loading of the corresponding capture register from the counter.

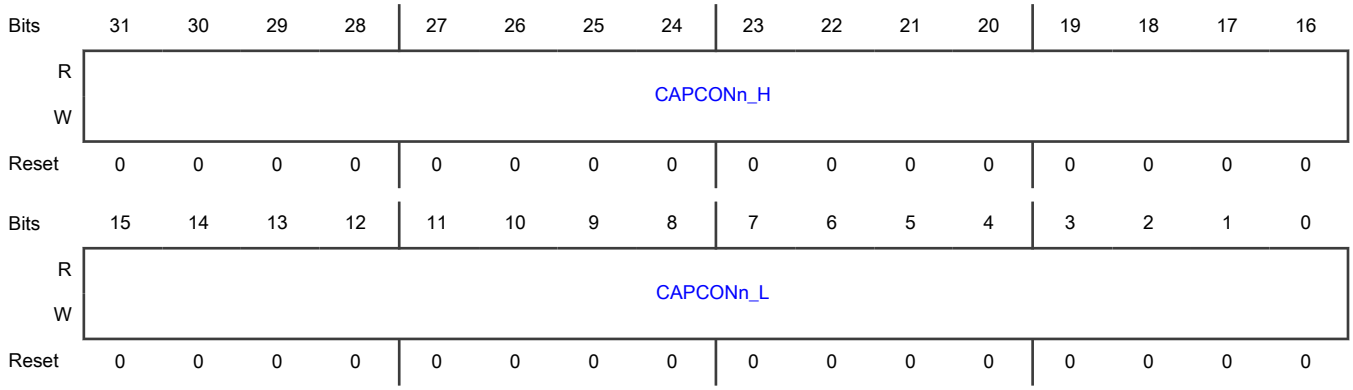
You can use this register depending on the configuration of CONFIG[UNIFY]:

- If CONFIG[UNIFY] = 1, only the _L fields are used.
- If CONFIG[UNIFY] = 0, you can write to this register as two registers, CAPCTRL*n*_L and CAPCTRL*n*_H. You can read both or write to the _L and _H registers individually.

Based on a selected event, the capture registers can be loaded with the current counter value when the event occurs.

Each capture control register (L, H, or unified 32-bit) controls which events cause the load of the corresponding capture register from the counter.

Diagram



Fields

Field	Function
31-16 CAPCONn_H	Capture Control High Captures higher bits. If bit *n* = 1, event *n* causes CAP*n*[CAP_H] (CONFIG[UNIFY] = 0) to be loaded (event 0 = bit 16, event 1 = bit 17, and so on). Number of bits = number of match and captures in SCT.
15-0 CAPCONn_L	Capture Control Low Captures lower bits. If bit *n* = 1, event *n* causes CAP*n*[CAP_L] (CONFIG[UNIFY] = 0) or CAP*n* (CONFIG[UNIFY] = 1) to be loaded (event 0 = bit 0, event 1 = bit 1, and so on). Number of bits = number of match and captures in SCT.

49.7.3.26 Match Reload Value (MATCHREL0 - MATCHREL15)

Offset

For n = 0 to 15:

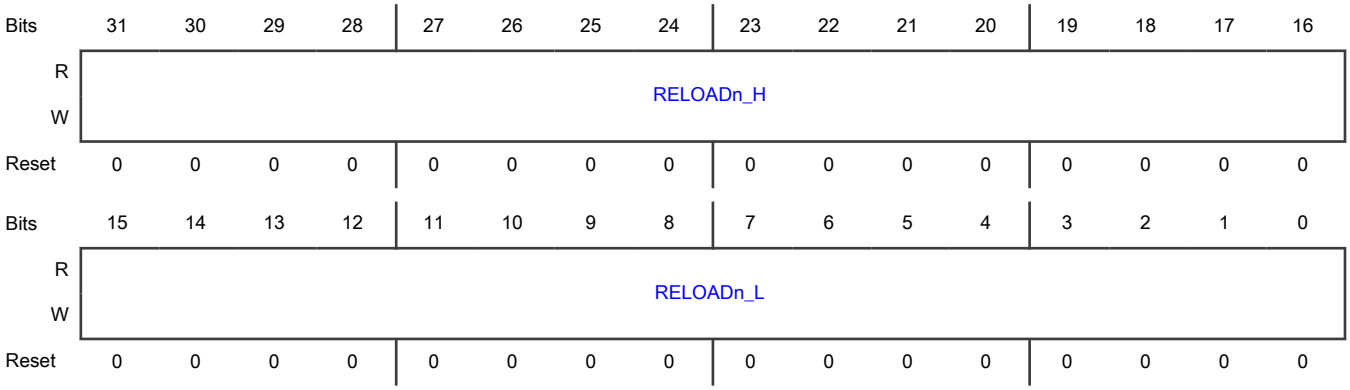
Register	Offset
MATCHRELn	200h + (n × 4h)

Function

Reloads a match register (L, H, or unified 32-bit counter) from its corresponding reload register at the start of each new counter cycle, which is when:

- CTRL[BIDIR_H] and CTRL[BIDIR_L] are 0 and the counter is cleared to zero upon reaching its limit condition.
- CTRL[BIDIR_H] and CTRL[BIDIR_L] are 1 and the counter counts down to 0, unless CONFIG[NORELOAD_H] and CONFIG[NORELOAD_L] are 1.

Diagram



Fields

Field	Function
31-16 RELOADn_H	Reload High Specifies the value to be loaded into MATCH*n*[MATCH_H]. If CONFIG[UNIFY] = 0, this field specifies the 16-bit value to be loaded into MATCH*n*[MATCH_H]. If CONFIG[UNIFY] = 1, the field specifies the upper 16 bits of the 32-bit value to be loaded into MATCH*n*[MATCH_H].
15-0 RELOADn_L	Reload Low Specifies the value to be loaded into MATCH*n*[MATCH_L]. If CONFIG[UNIFY] = 0, this field specifies the 16-bit value to be loaded into MATCH*n*[MATCH_L]. If CONFIG[UNIFY] = 1, the field specifies the lower 16 bits of the 32-bit value to be loaded into MATCH*n*[MATCH_L].

49.7.3.27 Fractional Match Reload (FRACMATREL0 - FRACMATREL5)

Offset

Register	Offset
FRACMATREL0	240h
FRACMATREL1	244h
FRACMATREL2	248h
FRACMATREL3	24Ch
FRACMATREL4	250h
FRACMATREL5	254h

Function

Reloads a fractional match register (_L, _H, or the unified 32-bit counter) from the corresponding fractional match reload register at the start of each new counter cycle.

A fractional match register is loaded from the corresponding fractional match reload register at the start of each new counter cycle, which means when:

- CTRL[BIDIR] = 0 and the counter is cleared to zero upon reaching its limit condition, or when CTRL[BIDIR] = 1 and the counter counts down to 0, unless the appropriate CONFIG[NORELOAD] field = 1.
- CTRL[BIDIR] = 1 and the counter is cleared to zero upon reaching its limit condition, or when CTRL[BIDIR] = 1 and the counter counts down to 0, unless the appropriate CONFIG[NORELOAD] field = 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												RELFRAC_H			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved												RELFRAC_L			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-16 RELFRAC_H	Reload Fractional Match High Specifies the value to be loaded into FRACMAT*n*[FRACMAT_H]. If CONFIG[UNIFY] = 0, read or write the 4-bit value to be loaded into FRACMAT*n*[FRACMAT_H]. If CONFIG[UNIFY] = 1, read or write the upper 4 bits with the 4-bit value to be loaded into FRACMAT*n*[FRACMAT_H].
15-4 —	Reserved
3-0 RELFRAC_L	Reload Fractional Match Low Specifies the value to be loaded into FRACMAT*n*[FRACMAT_L]. If CONFIG[UNIFY] = 0, read or write the 4-bit value to be loaded into FRACMAT*n*[FRACMAT_L]. If CONFIG[UNIFY] = 1, read or write the lower 4 bits to be loaded into FRACMAT*n*[FRACMAT_L].

49.7.3.28 Event n State (EV0_STATE - EV15_STATE)

Offset

For n = 0 to 15:

Register	Offset
EVn_STATE	300h + (n × 8h)

Function

Specifies event *n* state.

You can enable each event in some contexts (or states) and disable it in others. Each event defined in Event n Control (EV0_CTRL - EV15_CTRL) has an associated Event Interrupt Enable (EVEN) that can enable or disable the event for each available state.

Each event also has an associated Event n State (EV0_STATE - EV15_STATE) that allows this event to happen in one or more states of the counter selected by the corresponding EV*n*_CTRL[HEVENT] field.

An event *n* is disabled when its corresponding Event n State (EV0_STATE - EV15_STATE) contains all zeros because it is masked regardless of the current state.

In simple applications that do not use states, write 0x01 to this register to enable each event in exactly one state. Because the state does not change (which means, the state variable always remains at its reset value of 0), writing 0x01 permanently enables this event.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STATEMSKn															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STATEMSKn															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 STATEMSKn	Event State Mask Specifies the state in which event *n* happens. If bit *m* = 1, event *n* happens in state *m* of the counter selected by the HEVENT field (*n* = event number, *m* = state number; state 0 = bit 0, state 1= bit 1, and so on). Number of bits = number of states in this SCT.

49.7.3.29 Event n Control (EV0_CTRL - EV15_CTRL)

Offset

For n = 0 to 15:

Register	Offset
EVn_CTRL	304h + (n × 8h)

Function

Defines the conditions for an event to occur, based on the counter values or input and output states.

After you configure an event, it can be selected to trigger multiple actions (for example, stopping the counter and toggling an output) unless the event is blocked in the current SCT state, or the counter is halted. To block a particular event from occurring, use Event n State (EV0_STATE - EV15_STATE). To block all events for a given counter, write 1 to CTRL[HALT] or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value, matching a selected match register. In Bidirectional mode, you can enable events based on the direction of count.

If CONFIG[UNIFY] = 0, each event is associated with a particular counter in EV*n*_CTRL[HEVENT]. An event is permanently disabled when its event state mask register contains all 0s.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Only word writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved									DIRECTION		MATCHMEM	STATEV			
Reset	u	u	u	u	u	u	u	u	u	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STATEV	STATELD	COMBMODE		IOCOND		IOSEL				OUTSEL	HEVENT	MATCHSEL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-23 —	Reserved
22-21 DIRECTION	Direction Specifies the direction for event generation. This field only applies when the counters are operating in BIDIR mode. If CTRL[BIDIR] = 0, SCT ignores this field. 00b - Direction independent (event triggered regardless of the count direction)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Counting up (event triggered only during up-counting when CTRL[BIDIR] = 1) 10b - Counting down (event triggered only during down-counting when CTRL[BIDIR] = 1) 11b - Reserved
20 MATCHMEM	Match Mem If this field = 1 and EV*n*_CTRL[COMBMODE] specifies a match component to the triggering of this event, a match is considered to be active whenever the counter value is greater than or equal to the value specified in the match register when counting up, and less than or equal to the match value when counting down. If this field = 0, a match is only active during the cycle when the counter is equal to the match value.
19-15 STATEV	State Value Specifies the state value. The state value is loaded into or added to the state that EV*n*_CTRL[HEVENT] selects, depending on EV*n*_CTRL[STATELD], when this event is the highest-numbered event occurring for that state. If both the STATELD and STATEV fields are 0, there is no change to the value of STATE.
14 STATELD	State Load Controls how EV*n*_CTRL[STATEV] modifies the state that EV*n*_CTRL[HEVENT] selects when this event is the highest-numbered event occurring for that state. 0b - Value of STATEV added to that of STATE (the carry out is ignored) 1b - Value of STATEV loaded into that of STATE
13-12 COMBMODE	Combination Mode Selects how the specified match and I/O condition are used and combined. 00b - OR (the event occurs when either the specified match or I/O condition occurs) 01b - MATCH (uses the specified match only) 10b - IO (uses the specified I/O condition only) 11b - AND (the event occurs when the specified match and I/O condition occur simultaneously)
11-10 IOCOND	Input or Output Condition Selects the I/O condition for event *n*. The detection of edges on outputs lags the conditions that switch the outputs by one SCT clock. To guarantee proper edge and state detection, an input must have a minimum pulse width of at least one SCT clock period. 00b - Low 01b - Rise 10b - Fall 11b - High
9-6	Input or Output Signal Select

Table continues on the next page...

Table continued from the previous page...

Field	Function
IOSEL	Selects the input or output signal number associated with this event, if any. Do not select an input in this register if CONFIG[CLKMODE] is 1x. In this case, the clock input is an implicit ingredient of every event.
5 OUTSEL	Input and Output Select Selects the inputs or outputs that EV*n*_CTRL[IOSEL] specifies. 0b - Inputs 1b - Outputs
4 HEVENT	High Event Selects the low and high counters. Do not write 1 to this field if CONFIG[UNIFY] = 1. 0b - Low counter (selects the L state and the L match register that the MATCHSEL field specifies) 1b - High counter (selects the H state and the H match register that the MATCHSEL field specifies)
3-0 MATCHSEL	Match Select Selects the match register associated with this event, if any. A match can occur only when the counter that EV*n*_CTRL[HEVENT] selects is running.

49.7.3.30 Output n Set (OUT0_SET - OUT9_SET)

Offset

For n = 0 to 9:

Register	Offset
OUTn_SET	500h + (n × 8h)

Function

Selects the events that can set each SCT output.

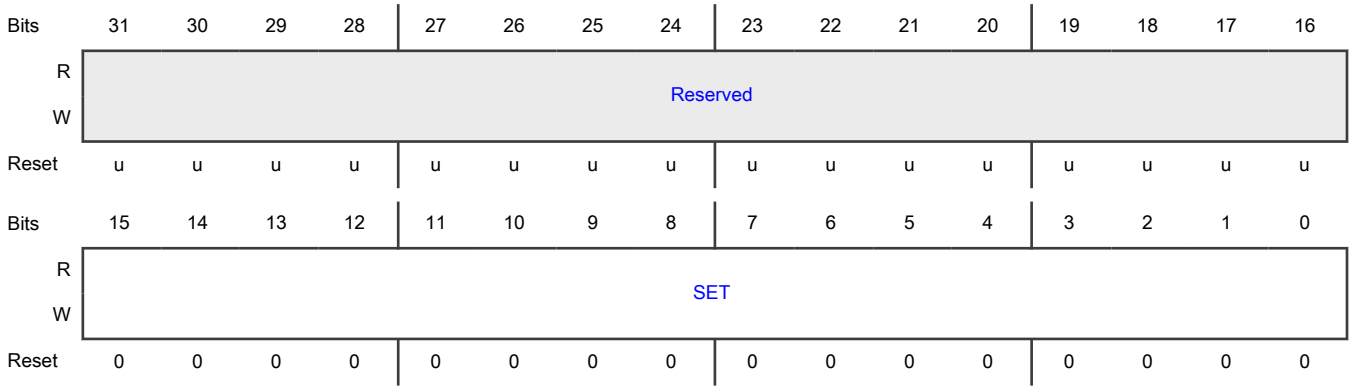
Based on a selected event, each SCT output can be set. There is one output set register for each SCT output. Each bit of an output set register is associated with a different event (bit 0 with event 0, and so on). The selected event can set or clear the output depending on the setting of OUTPUTDIRCTRL[SETCLR*n*]. To define the actual event that sets the output (a match, an I/O pin toggle, and so on), see Event n Control (EV0_CTRL - EV15_CTRL).

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

NOTE

If SCT is operating as two 16-bit counters, events can modify output states only when no counter is halted. This is true regardless of what triggered the event.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 SET	Set Output Selects events to set outputs. If bit *m* = 1, it selects event *m* to set output *n* (or clear it if OUTPUTDIRCTRL[SETCLR*n*] = 0x1 or 0x2). Output 0 = bit 0, output 1 = bit 1, and so on. Number of bits = number of events in this SCT. If the counter is used in Bidirectional mode, it is possible to reverse the action that the output set and clear registers specify when counting down (see Output Counter Direction Control (OUTPUTDIRCTRL) for more information).

49.7.3.31 Output n Clear (OUT0_CLR - OUT9_CLR)

Offset

For n = 0 to 9:

Register	Offset
OUTn_CLR	504h + (n × 8h)

Function

Selects the events that can clear each SCT output.

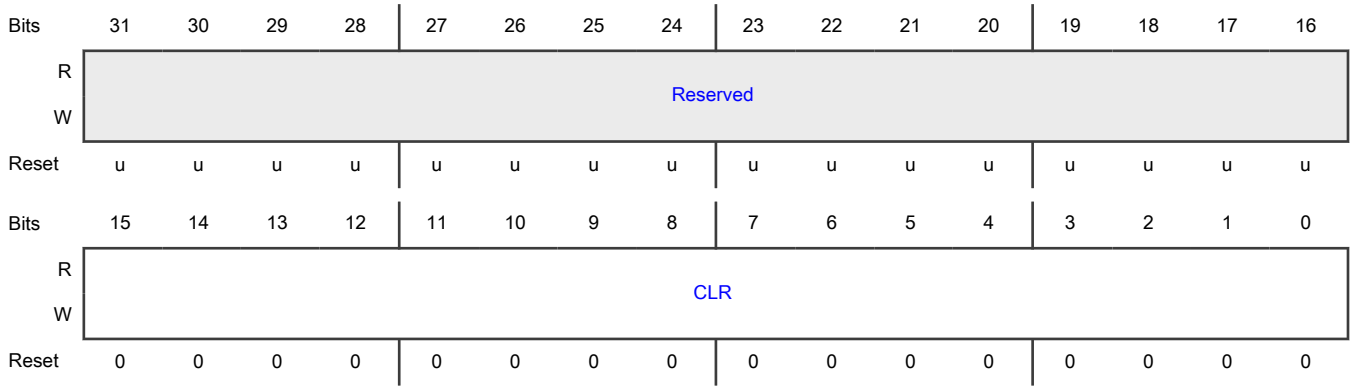
Based on a selected event, each SCT output can be cleared. There is one register for each SCT output that selects which events can clear that output. Each bit of an output clear register is associated with a different event (bit 0 with event 0, and so on). A selected event can clear or set the output depending on the setting of OUTPUTDIRCTRL[SETCLR*n*]. To define the actual event that clears the output (a match, an I/O pin toggle, and so on), see Event n Control (EV0_CTRL - EV15_CTRL).

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

NOTE

If SCT is operating as two 16-bit counters, events can modify output states only when no counter is halted. This is true regardless of what triggered the event.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CLR	Clear Output Selects events to clear outputs. If bit *m* = 1, it selects event *m* to clear output *n* (or set it if OUTPUTDIRCTRL[SETCLR*n*] = 0x1 or 0x2). Event 0 = bit 0; event 1 = bit 1, and so on. Number of bits = number of events in this SCT. If the counter is used in Bidirectional mode, it is possible to reverse the action specified by the output set and clear registers when counting down (see Output Counter Direction Control (OUTPUTDIRCTRL) for more information).

Chapter 50
Multi-Rate Timer (MRT)

50.1 Chip-specific MRT information

Table 435. Reference links to related information

Topic	Related module	Reference
Full description	MRT	MRT
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

50.1.1 Module instances

This device has one instance of the MRT module, MRT0.

50.1.2 Chip-specific MRT initialization

To initialize the MRT module:

1. Enable the clock to MRT via MRT field in AHB Clock control 1 AHBCLKCTRL1 register in SYSCON.
2. Deassert the MRT reset via MRT_RST field in the Peripheral reset control 1 PRESETCTRL1 register in SYSCON.
3. Ensure that the MRT can issue an interrupt in the chip. See registers for SmartDMA arch B inputs (SMARTDMAARCHB_INMUX0 - SMARTDMAARCHB_INMUX7) in INPUTMUX.

50.2 Overview

MRT provides a repetitive interrupt timer with 4 channels.

50.2.1 Block diagram

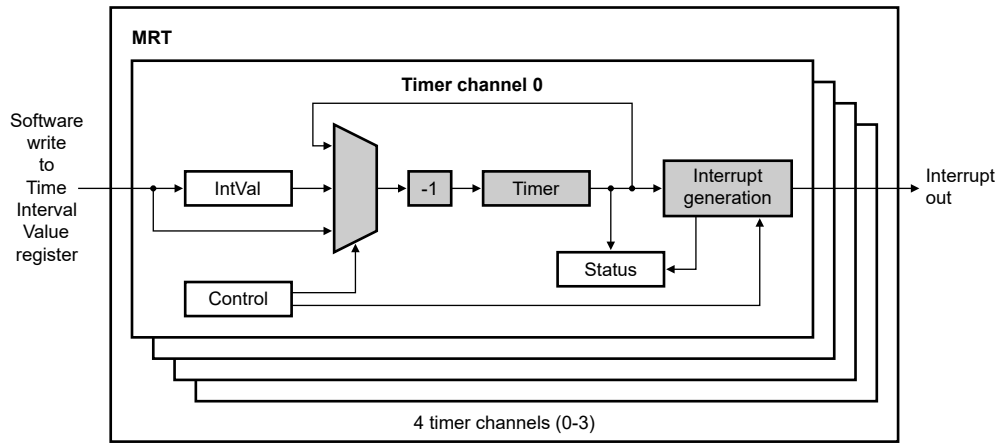


Figure 264. Block diagram

50.2.2 Features

- 24-bit interrupt timer.
- There are 4 channels independently counting down from individually set values.
- Repeat Interrupt, One-Shot Interrupt, and One-Shot Stall modes.
- You can program individual time intervals for each channel.
- Each channel operates independently of the other channels.

50.3 Functional description

The following sections discuss various modes that affect the functionality of MRT.

50.3.1 Operating modes

MRT has the following three operating modes:

- Repeat Interrupt mode
- One-Shot Interrupt mode
- One-Shot Stall mode

50.3.1.1 Repeat Interrupt mode

• Generates repeated interrupts after a selected time interval.

• Gets used for application-based Pulse Width Modulation (PWM) or Pulse Position Modulation (PPM).

When TIMER*n* is in idle state, writing a nonzero value (IVALUE) to INTVAL*n* immediately loads the time interval value (IVALUE -1). Then the timer begins to count down from this value.

After the timer reaches 0

- an interrupt is generated.
- the value in INTVAL*n* (IVALUE -1) is automatically reloaded, and
- the timer starts to count down again.

When the timer is running in Repeat Interrupt mode, you can perform various interval timer actions (change or stop):

Table 436. Possible actions in Repeat Interrupt mode

Action	To perform action	Timer does this	Interrupt
Change the interval value on the next interval timer cycle.	• Write new value (>0) to INTVAL*n*. • Write 0 to LOAD.	On the next cycle, the timer counts down from the new value.	An interrupt is generated when the timer reaches 0.
Change the interval value immediately (also called "change on-the-fly") in the interval timer.	• Write new value (>0) to INTVAL*n*. • Write 1 to LOAD.	The timer immediately starts to count down from the new timer interval value.	An interrupt is generated when the timer reaches 0.
Stop the interval timer at the end of the time interval.	• Write 0 to INTVAL*n*. • Write 0 to LOAD.	The timer stops at the end of the time interval.	An interrupt is generated when the timer reaches 0.
Stop the interval timer immediately.	• Write 0 to INTVAL*n*. • Write 1 to LOAD.	The timer stops immediately.	No interrupt is generated when INTVAL*n* is written.

50.3.1.2 One-Shot Interrupt mode

• In this mode, the timer generates one interrupt after a one-time count.

• This mode allows you to generate an interrupt at any time.

• Using this mode, you can introduce a specific delay in a software task.

When the timer is in an idle state, writing a nonzero value (IVALUE) to INTVAL*n* immediately loads the time interval value (IVALUE -1), and the timer starts to count down. As soon as the timer reaches 0, the timer stops and an interrupt is generated (and then enters the idle state).

When the timer is running in One-Shot Interrupt mode, you can perform the following interval timer actions (load or stop):

Table 437. Possible actions in One-Shot Interrupt mode

Action	To perform action	Timer does this	Interrupt
Load a timer interval value.	• Write a new time interval value (>0) to INTVAL*n*. • Write 1 to LOAD.	The timer immediately reloads the new time interval, and starts counting down from the new value.	An interrupt is generated when the timer reaches 0.
Stop the interval timer.	• Write a 0 to INTVAL*n*. • Write 1 to LOAD.	The timer immediately stops counting and moves to the idle state.	No interrupt is generated when INTVAL*n* is updated.

50.3.1.3 One-Shot Stall mode

• Used for very short delays (for example, when the delay needed is less than the time it takes to arrive at an interrupt service routine).

• Designed for very low software overhead that requires only a single write to INTVAL*n* (if the channel is already configured for One-Shot stall mode). MRT counts the requested delay when stalling the bus write operation, and then concluding the write operation when the delay has finished. No interrupt or status polling is needed.

Use this mode:

- when a short delay is needed between two software-controlled events.
- when a delay is expected before your application can continue.

The CPU consumes a minimum amount of power during One-Shot Stall mode, because no bus transactions occur during that period.

One-Shot Stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL*n* and the time that your software resumes execution. Other system events, such as interrupts or other bus masters accessing the APB bus where MRT resides, can cause the delay to be longer.

50.3.2 Clocking

Table 438. MRT clocks

Type of clock	Description
Bus clock (vpb_clk)	Controls the access to the MRT registers and functions the MRT module.

50.3.3 Interrupts

A channel provides 3 different interrupts modes listed below and can select the interrupt mode individually per channel. This interrupt can be programmed with an independent time interval.

- Repeat interrupt

- Shot interrupt
- Shot bus stall interrupt

50.4 External signals

This module has no external signals.

50.5 Initialization

To initialize:

1. Enable the clock to MRT.
2. Deassert MRT reset.
3. Ensure that MRT can issue an interrupt in the chip.

See the Chip-specific section for more details.

50.6 Memory map and register descriptions

50.6.1 Multi-Rate Timer (MRT) register descriptions

50.6.1.1 MRT memory map

MRT0 base address: 4001_3000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Time Interval Value (INTVAL0)	32	RW	See section
4h	Timer (TIMER0)	32	R	See section
8h	Control (CTRL0)	32	RW	See section
Ch	Status (STAT0)	32	RW	See section
10h	Time Interval Value (INTVAL1)	32	RW	See section
14h	Timer (TIMER1)	32	R	See section
18h	Control (CTRL1)	32	RW	See section
1Ch	Status (STAT1)	32	RW	See section
20h	Time Interval Value (INTVAL2)	32	RW	See section
24h	Timer (TIMER2)	32	R	See section
28h	Control (CTRL2)	32	RW	See section
2Ch	Status (STAT2)	32	RW	See section
30h	Time Interval Value (INTVAL3)	32	RW	See section
34h	Timer (TIMER3)	32	R	See section
38h	Control (CTRL3)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
3Ch	Status (STAT3)	32	RW	See section
F0h	Module Configuration (MODCFG)	32	RW	See section
F4h	Idle Channel (IDLE_CH)	32	R	See section
F8h	Global Interrupt Flag (IRQ_FLAG)	32	RW	See section

50.6.1.2 Time Interval Value (INTVAL0 - INTVAL3)

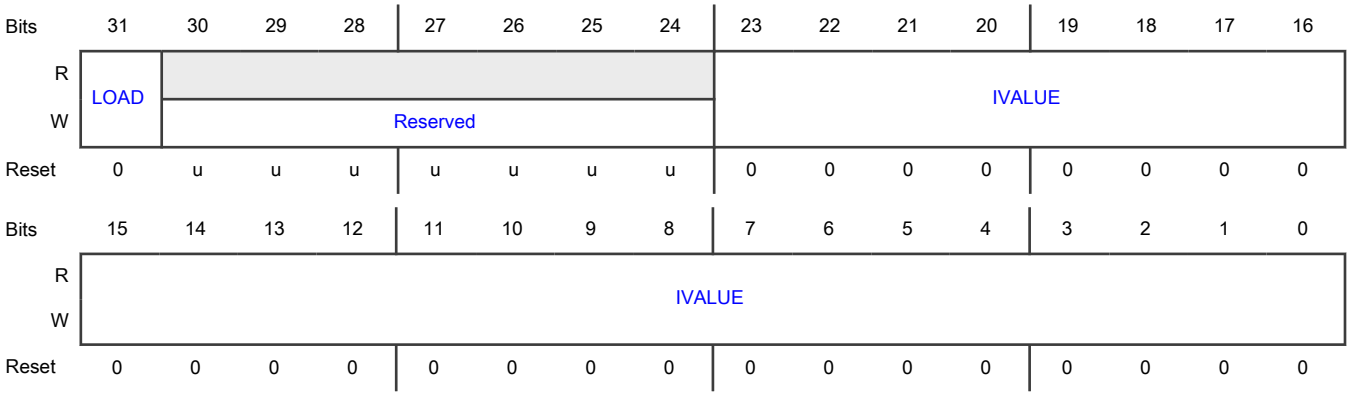
Offset

Register	Offset
INTVAL0	0h
INTVAL1	10h
INTVAL2	20h
INTVAL3	30h

Function

Contains MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Diagram



Fields

Field	Function
31 LOAD	Force Load Enable Determines how the timer interval value (IVALUE -1) is loaded into the TIMER*n*.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	LOAD is write only. Reading LOAD always returns 0. If the repeat mode is selected, the load from the INTVAL*n* to the TIMER*n* is processed at the end of the time interval. The INTVAL*n* interval value (IVALUE -1) is immediately loaded into the TIMER*n* when it is running. 0b - No force load 1b - Force load
30-24 —	Reserved Reserved. Read value is undefined; only 0 should be written.
23-0 IVALUE	Time Interval Load Value. INTVAL*n*[IVALUE] is loaded into TIMER*n*, and MRT channel *n* starts counting down from IVALUE -1. If the timer is idle, then writing a nonzero value to INTVAL*n*[IVALUE] starts the timer immediately. If the timer is running, then writing a 0 to INTVAL*n*[IVALUE] does the following: if LOAD = 1, then the timer stops immediately; if LOAD = 0, then the timer stops at the end of the time interval.

50.6.1.3 Timer (TIMER0 - TIMER3)

Offset

Register	Offset
TIMER0	4h
TIMER1	14h
TIMER2	24h
TIMER3	34h

Function

Holds the current timer value (the TIMER*n* reads the value of the down counter). TIMER*n* registers are read only.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								VALUE							
W																
Reset	u	u	u	u	u	u	u	u	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-24 —	Reserved Reserved. Read value is undefined; only 0 should be written.
23-0 VALUE	Current Timer Value Holds the current timer value of the down counter. The initial value of TIMER*n* is loaded as (IVALUE -1) from INTVAL*n*, either at the end of the time interval or immediately in the following cases: INTVAL*n* is updated in the idle state or INTVAL*n* is updated with LOAD = 1. When the timer is in an idle state, reading TIMER*n*[VALUE] fields returns -1 (00FF FFFFh).

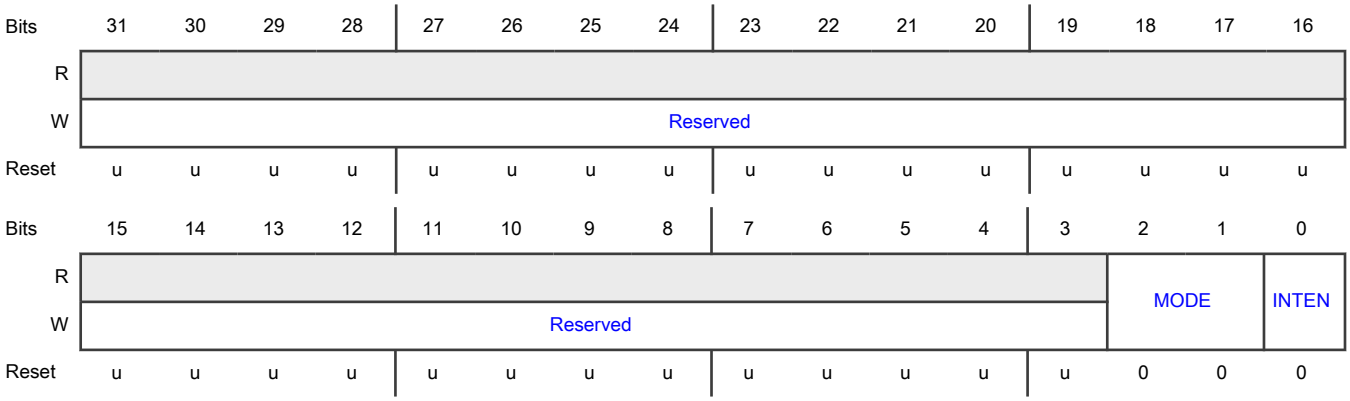
50.6.1.4 Control (CTRL0 - CTRL3)

Offset

Register	Offset
CTRL0	8h
CTRL1	18h
CTRL2	28h
CTRL3	38h

Function
Controls the modes and enables the interrupt for each MRT.

Diagram



Fields

Field	Function
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2-1 MODE	MRT Operating mode Selects the timer mode. 00b - Repeat Interrupt mode 01b - One-Shot Interrupt mode 10b - One-Shot Stall mode 11b - Reserved
0 INTEN	Interrupt request Enables the TIMER*n* interrupt. 0b - Disabled 1b - Enabled

50.6.1.5 Status (STAT0 - STAT3)

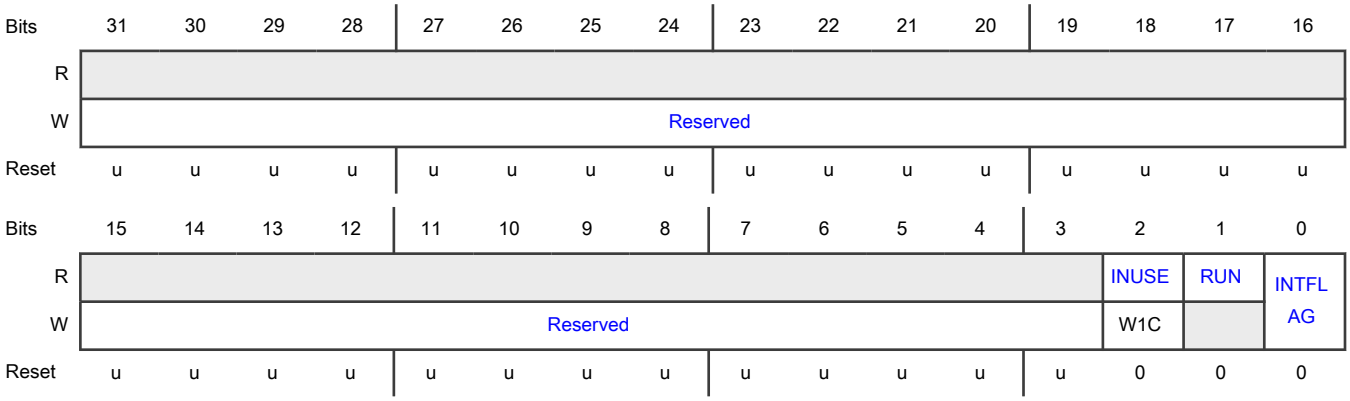
Offset

Register	Offset
STAT0	Ch
STAT1	1Ch
STAT2	2Ch
STAT3	3Ch

Function

Indicates the status of each MRT timer channel.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 INUSE	Channel-In-Use flag Affects the use of Idle Channel (IDLE_CH); the operating details depend on the operating mode bit in MODCFG[MULTITASK]. Writing 1 to this bit clears the status. 0b - This timer channel is not in use. 1b - This timer channel is in use.
1 RUN	Timer n State Indicates the state of TIMER*n*. STAT*n*[RUN] is read only. TIMER*n* stops in idle state and runs in running state. 0b - Idle state. 1b - Running.
0 INTFLAG	Interrupt Flag Monitors the interrupt flag. If the INTEN bit in the CONTROLn is also set to 1, then the interrupt for timer channel n and the global interrupt are generated. Writing a 1 to INTFLAG bit clears the interrupt request. 0b - No pending interrupt. 1b - Pending interrupt.

50.6.1.6 Module Configuration (MODCFG)

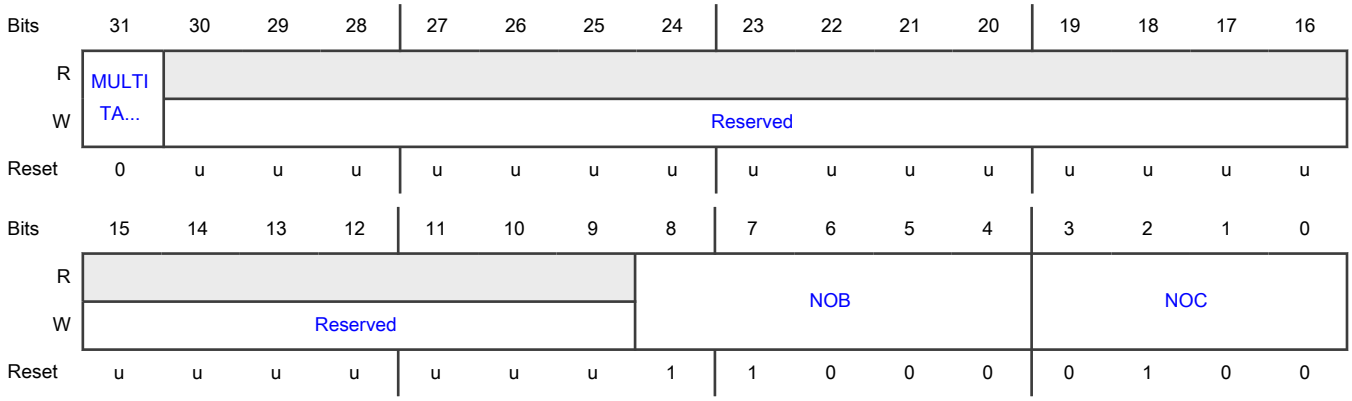
Offset

Register	Offset
MODCFG	F0h

Function

Provides information about this particular MRT instance. It also allows selecting an overall mode for the idle channel feature. MODCFG[MULTITASK] provides the configuration (number of channels and timer width) for this MRT. A reset of the INUSE(n) flags for all channels happens in Hardware Status mode.

Diagram



Fields

Field	Function
31 MULTITASK	MULTITASK Selects the operating mode for STAT*n*[INUSE] and Idle Channel (IDLE_CH). 0b - Hardware status mode. 1b - Multitask mode
30-9 —	Reserved Read value is undefined; only 0 should be written.
8-4 NOB	Number of Bits Identifies the number of timer bits in this MRT (24 bits on this chip).
3-0 NOC	Number of Channels Identifies the number of channels in this MRT.

50.6.1.7 Idle Channel (IDLE_CH)

Offset

Register	Offset
IDLE_CH	F4h

Function

Returns the number of the first Idle Channel.

Use Idle Channel (IDLE_CH) for finding available channels in MRT. This allows you more flexibility when using MRT—no fixed channel assignments are required, and you do not need to search for an available channel. Generally, Idle Channel (IDLE_CH) returns the lowest available channel number.

MODCFG[MULTITASK] allows you to use Idle Channel (IDLE_CH) in two ways. MODCFG[MULTITASK] affects both the functions of Idle Channel (IDLE_CH), and the function of STAT*n*[INUSE] for each MRT channel:

- MULTITASK = 0: Hardware status mode. The STAT*n*[INUSE] for all MRT channels are reset. Idle Channel (IDLE_CH) returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: Multitask mode. In Multitask mode, the STAT*n*[INUSE] allow more control over when MRT channels are released for further use. When Idle Channel (IDLE_CH) is read, returning a channel number of an idle channel, the STAT*n*[INUSE] for that channel is set by hardware. That channel is not considered idle until its RUN flag = 0, and there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and there is no need to start the channel before that channel is no longer considered idle by Idle Channel (IDLE_CH). It also allows you to identify a specific MRT channel. Then you can use that channel more than once without releasing it, removing the need to ask for an available channel for every use.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								CHAN				Reserved			
W																
Reset	u	u	u	u	u	u	u	u	0	0	0	0	u	u	u	u

Fields

Field	Function
31-8 —	Reserved
7-4 CHAN	Idle Channel Helps in reading IDLE_CH[CHAN] and returns the lowest idle timer channel. The number is positioned so that you can use it as an offset from MRT base address. This provides access to the registers for the allocated channel. If all timer channels are running, then CHAN = Fh.
3-0 —	Reserved

50.6.1.8 Global Interrupt Flag (IRQ_FLAG)

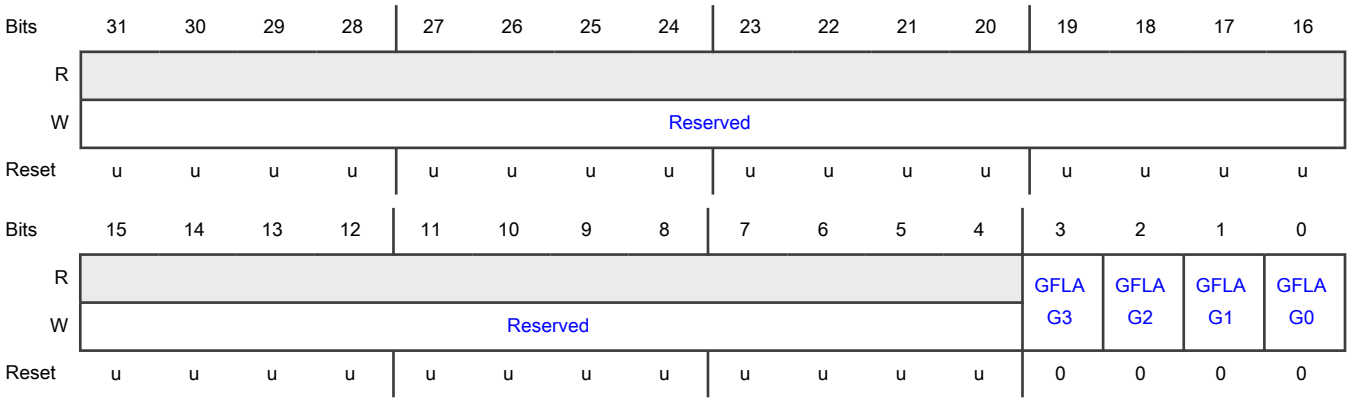
Offset

Register	Offset
IRQ_FLAG	F8h

Function

Combines the interrupt flags from the individual timer channels into one register. Setting and clearing each flag behaves in the same way as setting and clearing STAT*n*[INTFLAG] in each of the STATUS*n* registers.

Diagram



Fields

Field	Function
31-4 —	Interrupt Flag Reserved. Read value is undefined; only 0 should be written.
3 GFLAG3	Interrupt Flag Monitors the interrupt flag of TIMER3 and acts similar to channel 0.
2 GFLAG2	Interrupt Flag Monitors the interrupt flag of TIMER2 and acts similar to channel 0.
1 GFLAG1	Interrupt Flag Monitors the interrupt flag of TIMER1 and acts similar to channel 0.
0 GFLAG0	Interrupt Flag Monitors the interrupt flag of TIMER0. Writing 0 is equivalent to no operation. In the pending interrupt state, Timer (TIMER0 - TIMER3) has reached the end of the time interval. If CTRL*n*[INTEN] in CONTROL*n* also gets 1, then there is a timer channel *n* interrupt and a global interrupt. Writing 1 to IRQ_FLAG[GFLAG0] clears the interrupt request. 0b - No pending interrupt. 1b - Pending interrupt

Chapter 51
Windowed Watchdog Timer (WWDT)

51.1 Chip-specific WWDT information

Table 439. Reference links to related information

Topic	Related module	Reference
Full description	WWDT	WWDT
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management

NOTE

A watchdog reset resets the chip. See WWDT for details.

51.1.1 Module instances

This device has two instances of the WWDT module, WWDT0 and WWDT1.

51.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

51.1.3 MOD[LOCK] behavior

When MOD[LOCK] is set, the write accesses to SYSCON registers are blocked.

- WWDT0: After MOD[LOCK] is set to one and a Watchdog feed is performed, hardware prevents changes to the Watchdog Clock Divider,WDT0 Clock Divider (WDT0CLKDIV), register. MOD[LOCK] can be set once by software and is only cleared by any reset.
- WWDT1: After MOD[LOCK] is set to one and a Watchdog feed is performed, hardware prevents changes to the Watchdog clock divider, WDT1 Function Clock Divider (WDT1CLKDIV), register or WDT1 clock mux select, WDT1 Clock Selection (WDT1CLKSEL), register. MOD[LOCK] can be set once by software and is only cleared by any reset.

51.2 Overview

WWDT helps you reset or interrupt a microcontroller within a programmable time, if the microcontroller (or core) is stuck in an infinite loop or is executing an unintended code. If an application fails to reload or feed a watchdog (WDOG) timer within the predefined duration of time, it generates a watchdog reset (if enabled).

51.2.1 Block diagram

In the following WWDT block diagram, the synchronization logic (APB bus clock to WDCLK) is not shown. See the chip-specific WWDT information section or clocking chapter for more details.

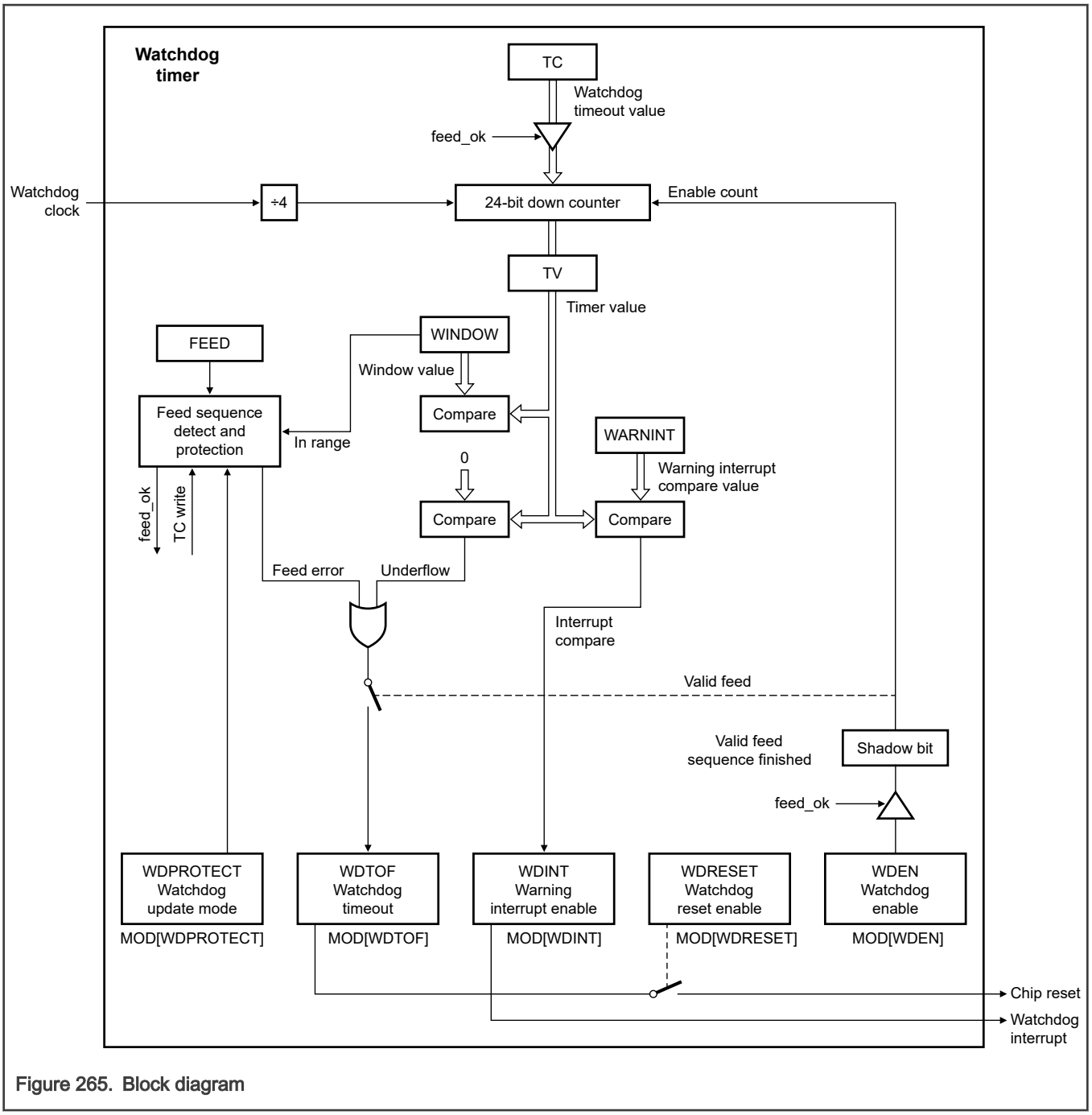


Figure 265. Block diagram

51.2.2 Features

- Can reset or interrupt the microcontroller within a programmable time
- Watchdog reset indication flag
- Clocking features:
  - Programmable 24-bit timer with an internal fixed (divided by 4) prescaler
  - Selectable time period in increments of four watchdog clocks: From 1024 watchdog clocks (T_{WDCLK} x 256 x 4) to over 67 million watchdog clocks (T_{WDCLK} x 2^{24} x 4)

  - Watchdog clock (WDCLK) source having a selectable frequency in the range of 6 kHz to 1.5 MHz (see the chip-specific information section and the Clocking chapter for detailed setting in this device)
- Can be configured to run in Low-Power mode (see the chip-specific information section and the Power Management chapter for detailed setting in this device)
- Other configurable features:
  - Windowed operation requiring a watchdog reload to occur between a programmable minimum and maximum timeout period
  - Generation of a warning interrupt at a programmable time before watchdog timeout
  - Protection of the watchdog reload value so that you can change it only after the warning interrupt time is reached

51.3 Functional description

When you program a watchdog window (a timing window), an early watchdog reload is also treated as a watchdog event, which prevents situations where a system failure may still reload the watchdog timer. For example, the application code could be stuck in an interrupt service that contains a watchdog reload (feed). You must set the timing window in a way that this situation causes an early watchdog timer reload, which generates a watchdog event, and then enables the system to recover from this situation.

WWDT consists of a fixed (divided by 4) prescaler and a 24-bit counter. The minimum value of the counter is FFh; if you write a value lower than FFh, the counter automatically loads with the value FFh.

- Minimum watchdog interval is (T_{WDCLK} X 256 X 4).
- Maximum watchdog interval is (T_{WDCLK} X 2^{24} X 4) in multiples of (T_{WDCLK} X 4).

51.3.1 Operating modes

This section describes all functional operating modes of the WWDT module: Disabled, Watchdog Interrupt, and Watchdog Reset.

Table 440. Watchdog operating modes selection

MOD[WDEN]	MOD[WDRESET]	Mode	Mode description
0	X (0 or 1)	Disabled	Debugs or operates without the watchdog running.
1	0	Watchdog Interrupt	Generates a watchdog warning interrupt but does not enable the watchdog reset. A watchdog counter equal to the value of WARNINT[WARNINT] sets MOD[WDINT] and generates a watchdog interrupt request.
1	1	Watchdog Reset	Enables both the watchdog interrupt and watchdog reset: • A watchdog counter equal to the value of WARNINT[WARNINT] sets MOD[WDINT] and generates a watchdog interrupt request. • A watchdog counter equal to 0 resets the microcontroller. • A watchdog feed before reaching the value of WINDOW[WINDOW] also causes a watchdog reset.

51.3.2 Example timing diagrams to show WWDT in operation

A feed is correct when both of the following conditions are true:

- A valid feed sequence completes, writing AAh followed by 55h to FEED[FEED].
- The value of TV[COUNT] is not greater than the value of WINDOW[WINDOW].

If either of these conditions is not true, a feed error occurs.

In the correct watchdog feed with Windowed mode enabled, the sequence of writing AAh followed by 55h occurs when the counter value (11FCh) is not greater than the value of WINDOW[WINDOW] (1200h). It is a correct feed (reload) event. After the correct feed event occurs, the watchdog counter reloads with the TC[COUNT] value (2000h).

In addition to the above two conditions, when MOD[WDPROTECT] = 1, the watchdog counter must not be greater than the value of WARNINT[WARNINT] for a correct feed to occur. See the following figure for a pictorial representation.

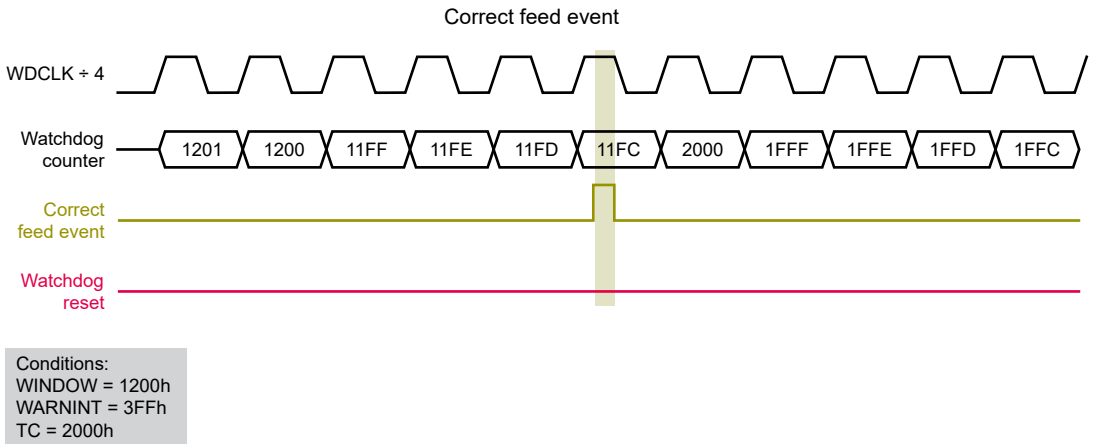


Figure 266. Correct watchdog feed with Windowed mode enabled

A correct feed sequence can only occur when the value of TV[COUNT] is not greater than the value of WINDOW[WINDOW] (1200h). Otherwise, a feed error occurs.

The following figure shows a reset trigger when you feed the watchdog too early.

An early feed event occurs if the feed sequence occurs when the watchdog counter value (1257h) is greater than the value of WINDOW[WINDOW] (1200h). This feed error can generate a reset if MOD[WDRESET] = 1.

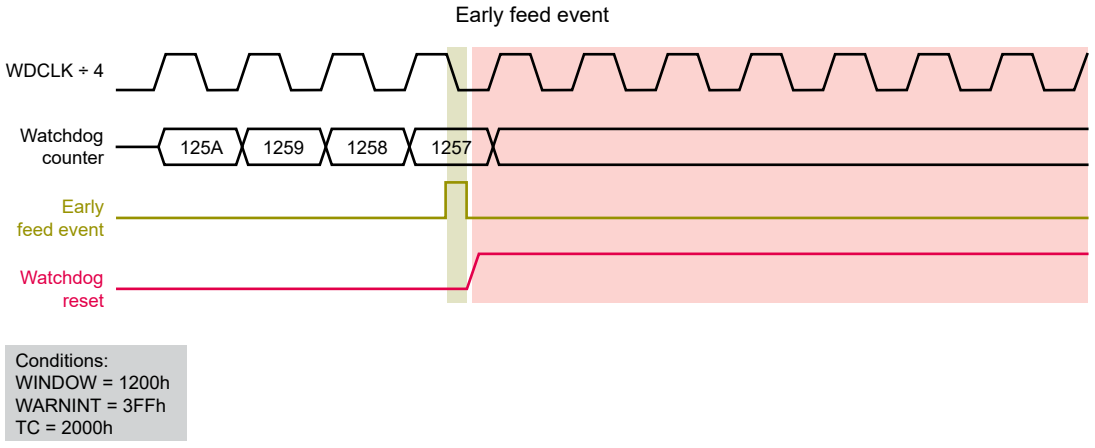


Figure 267. Early watchdog feed with Windowed mode enabled

The following figure shows WWDT generating a warning.

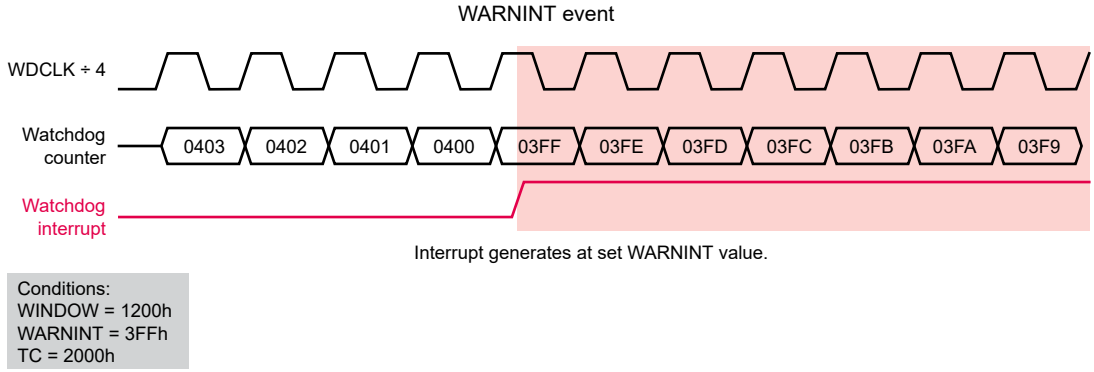


Figure 268. Watchdog warning interrupt

51.3.3 Clocking

WWDT uses two clocks: APB bus clock and WDCLK.

- The system clock derives the APB bus clock that is used for APB accesses to the watchdog registers.
- The watchdog oscillator derives WDCLK that is used for watchdog timer counting.

The synchronization logic between the two clock domains works as follows:

- When updating Mode (MOD) and Timer Constant (TC), the new value takes effect in three WDCLK cycles on the logic in the WDCLK clock domain.
- When the watchdog timer is counting on WDCLK, the synchronization logic first locks the value of the counter on WDCLK and then synchronizes it with the APB bus clock, so that the CPU can read Timer Value (TV).

NOTE

Because of the synchronization step, you must add a delay of three WDCLK clock cycles between the feed sequence and the time that you take to enable the functionality of MOD[WDPROTECT]. The length of the delay depends on the selected watchdog clock, WDCLK.

51.3.4 Using the WWDT lock

WWDT supports lock features to ensure continuous operation. These features prevent:

- The disabling of the WWDT clock source.
- The changing of the WWDT reload value.

51.3.4.1 Preventing the disabling of the WWDT clock source

If MOD[LOCK] = 1, the WWDT clock source is locked. You cannot disable the clock source when entering Sleep or Deep-Sleep modes. Therefore, enable the watchdog oscillator for each Power mode before writing 1 to MOD[LOCK].

51.3.4.2 Preventing the changing of the WWDT reload value

If MOD[WDPROTECT] = 1, then any attempt to change the value of TC[COUNT] with a feed sequence, before the watchdog counter value becomes less than the values of WARNINT[WARNINT] and WINDOW[WINDOW], sets MOD[WDTOF] and causes a watchdog reset.

Any type of reset disables the reload overwrite lock mechanism.

51.4 External signals

This module has no external signals.

51.5 Initialization

Perform the following steps to initialize WWDT:

1. Enable and configure the watchdog oscillator.
2. Set the watchdog timer constant reload value using TC[COUNT].
3. Set the watchdog timer operating mode using MOD[WDPROTECT].
4. Set a value for the watchdog window time using WINDOW[WINDOW], if the windowed operation is desired.
5. Set a value for the watchdog warning interrupt using WARNINT[WARNINT] and configure the system interrupt controller, if a warning interrupt is desired.
6. Enable the watchdog by writing AAh followed by 55h to FEED[FEED].

To prevent a watchdog event, you must feed or reload WWDT again before the watchdog counter reaches 0. If you set a value for the watchdog window time, then the feed or reload must also occur after the watchdog counter passes the value of WINDOW[WINDOW].

After you configure WWDT in a way that a watchdog event causes a reset (MOD[WDRESET] = 1), or when the counter reaches 0 (it causes a reset), then the microcontroller (or core) is also reset (loading the stack pointer and program counter from the vector table, just like the way it happens for an external reset).

51.6 Memory map and register definition

This section includes the WWDT module memory map and detailed descriptions of all registers.

51.6.1 WWDT register descriptions

51.6.1.1 WWDT memory map

WWDT0 base address: 4001_6000h

WWDT1 base address: 4001_7000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Mode (MOD)	32	RW	0000_0000h
4h	Timer Constant (TC)	32	RW	0000_00FFh
8h	Feed Sequence (FEED)	32	W	See section
Ch	Timer Value (TV)	32	R	0000_00FFh
14h	Warning Interrupt Compare Value (WARNINT)	32	RW	0000_0000h
18h	Window Compare Value (WINDOW)	32	RW	00FF_FFFFh

51.6.1.2 Mode (MOD)

Offset

Register	Offset
MOD	0h

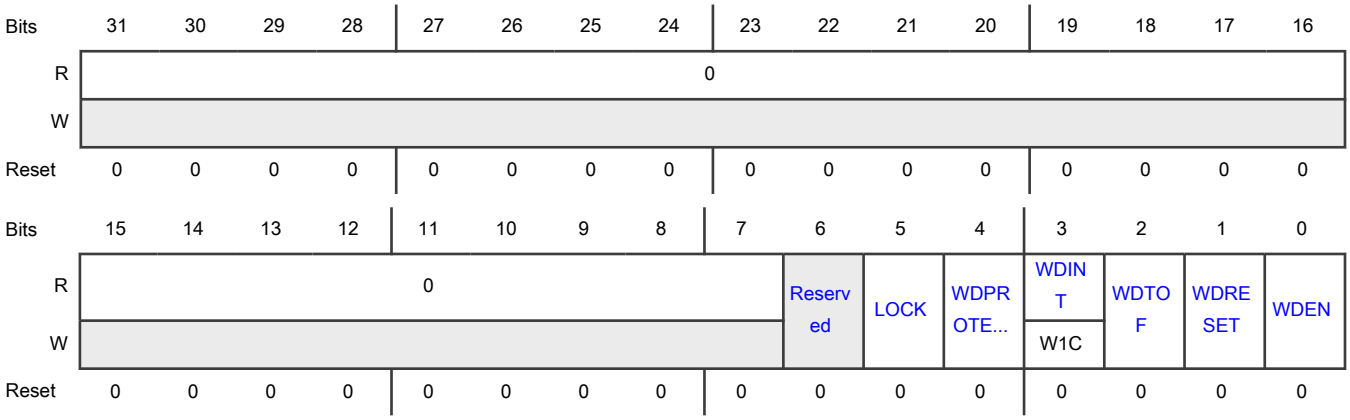
Function

Controls the operation of WWDT. You must perform a watchdog feed before any changes to this register take effect.

NOTE

After you write 1 to MOD[WDEN], MOD[WDPROTECT], or MOD[WDRESET], you can return those fields to 0 only with an external reset or a WWDT reset. Until those resets occur, writing 0 to the aforementioned fields has no effect.

Diagram



Fields

Field	Function
31-7 —	Reserved
6 —	Reserved
5 LOCK	Lock Prevents disabling or powering down of the watchdog oscillator after you write 1 to this field and perform a watchdog feed. You can write 1 to this field, which can become 0 only with a reset. 0b - No Lock 1b - Lock
4 WDPROTECT	Watchdog Update Mode Protects the value of TC[COUNT] from changing at any time. In Flexible mode (when this field is 0), you can change the value of TC[COUNT] at any time. In Threshold mode (when this field is 1), you can change the value of TC[COUNT] only after the counter is less than the individual values of WARNINT[WARNINT] and WINDOW[WINDOW]. Any attempt to change the value of TC[COUNT] with a feed sequence, before the watchdog counter value becomes less than the values of WARNINT[WARNINT] and WINDOW[WINDOW], sets MOD[WDTOF] and causes a watchdog reset. You can write 1 to this field, which can become 0 only with a reset (see the note in MOD register).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Flexible 1b - Threshold
3 WDINT	Warning Interrupt Flag Sets the flag when the timer is at or below the value defined in WARNINT[WARNINT]. Note that you cannot clear this flag while the value of WARNINT[WARNINT] is equal to the value of TV[COUNT]. This can occur if the value of both WARNINT[WARNINT] and MOD[WDRESET] is 0 when TV[COUNT] decrements to 0. When you configure WWDT to generate a warning interrupt, the interrupt occurs when the counter is no longer greater than the value defined by WARNINT[WARNINT]. If a feed sequence occurs when the watchdog counter value is greater than the value of WINDOW[WINDOW], this flag is set. Any reset clears this flag. 0b - No flag 1b - Flag
2 WDTOF	Watchdog Timeout Flag The bit could be set if: • Watchdog timer timeout has occurred • Feed error has occurred You must clear this flag by writing 0 to it. An external reset or POR clears this flag too. 0b - Watchdog event has not occurred. 1b - Watchdog event has occurred (causes a chip reset if WDRESET = 1).
1 WDRESET	Watchdog Reset Enable Allows a watchdog timeout to cause a chip reset, if this field = 1. After you write 1 to this field, you cannot write 0 to it (see the note in MOD register). 0b - Interrupt 1b - Reset
0 WDEN	Watchdog Enable Enables or disables the watchdog timer. After writing 1 to this field, perform a watchdog feed (reload) to enable the watchdog timer. After you write 1 to this field, you cannot write 0 to it (see the note in MOD register). 0b - Timer stopped 1b - Timer running

51.6.1.3 Timer Constant (TC)

Offset

Register	Offset
TC	4h

Function

Specifies the desired timeout value. Transferring the value of this register into the watchdog counter requires a feed sequence. The register resets to 00_00FFh. Writing a value below FFh causes 00_00FFh to load into the register. Therefore, the minimum timeout interval is T_{WDCLK} X 256 X 4.

If MOD[WDPROTECT] = 1, then any attempt to change the value of this register with a feed sequence, before the watchdog counter value becomes less than the individual values of WARNINT[WARNINT] and WINDOW[WINDOW], sets MOD[WDTOF] and causes a watchdog reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								COUNT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Fields

Field	Function
31-24 —	Reserved
23-0 COUNT	Watchdog Timeout Value Specifies the desired timeout value for WWDT.

51.6.1.4 Feed Sequence (FEED)

Offset

Register	Offset
FEED	8h

Function

Reloads the watchdog timer with the TC[COUNT] value, if you write AAh followed by 55h to FEED[FEED]. This operation also starts WWDT, if the watchdog timer is enabled by writing 1 to MOD[WDEN].

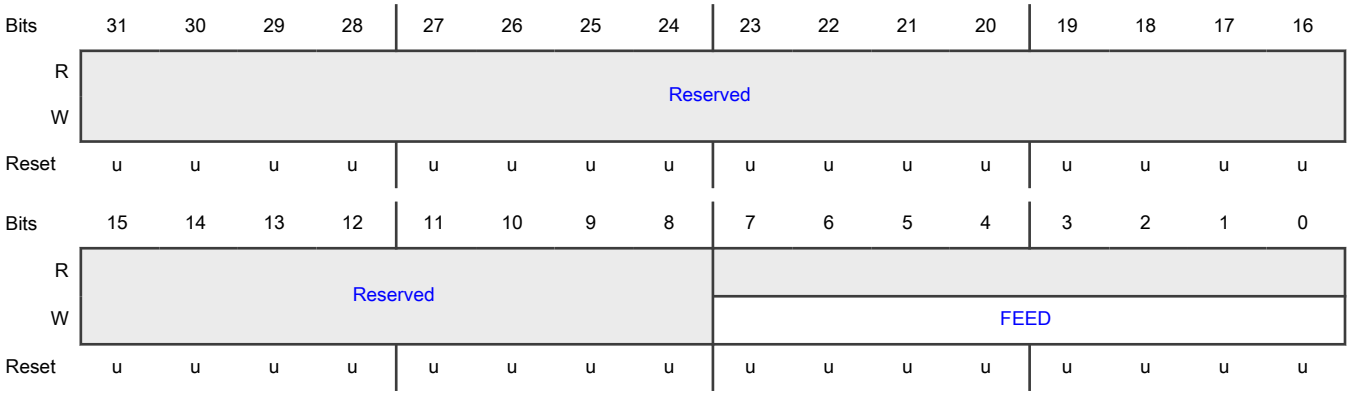
However, writing 1 to MOD[WDEN] is not sufficient to enable WWDT. Before the watchdog can generate a reset, MOD[WDEN] must be 1, followed by a valid feed sequence that completes itself. Until then, the watchdog ignores feed errors.

After writing AAh to FEED[FEED], access to any watchdog register other than writing 55h to FEED[FEED] causes an immediate reset or interrupt when WWDT is enabled and sets MOD[WDTOF]. An incorrect access to a watchdog register during a feed sequence generates the reset during the second APB bus clock.

To avoid an unintended interrupt, it is a good practice to disable interrupts around a feed sequence. Disable interrupts if they may result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to WWDT before control returns to the interrupted task.

If the value of WINDOW[WINDOW] is less than the default value, it may limit the time for which a watchdog feed is allowed.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 FEED	Feed Value Specifies feed value. The value must be AAh followed by 55h.

51.6.1.5 Timer Value (TV)

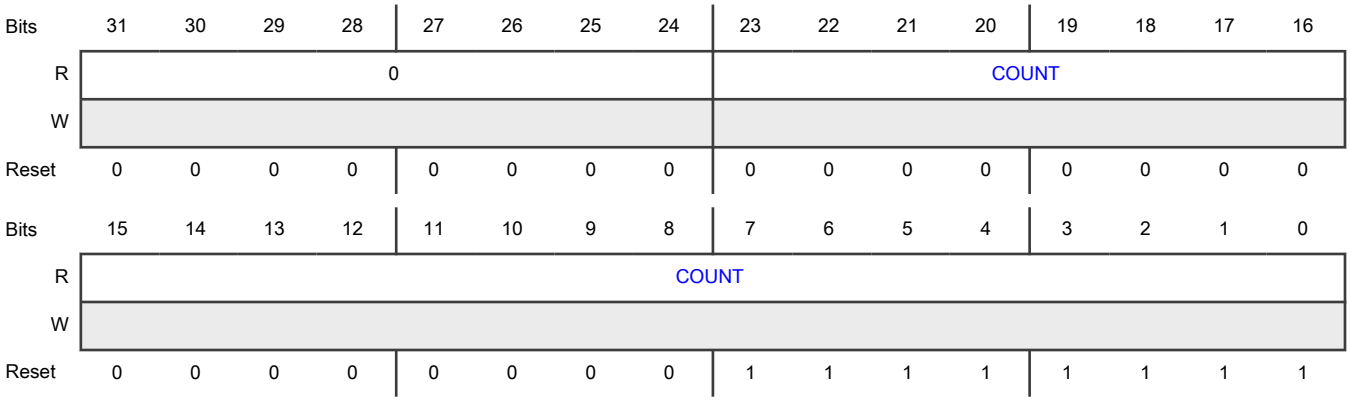
Offset

Register	Offset
TV	Ch

Function

Contains the current value of the WWDT counter. When the core (or CPU) reads this register, the value of this register is older than the actual value of the timer. Reading the timer using the lock and synchronization procedure takes up to six WDCLK cycles plus six APB bus clock cycles.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 COUNT	Counter Timer Value Specifies the current value of the watchdog timer.

51.6.1.6 Warning Interrupt Compare Value (WARNINT)

Offset

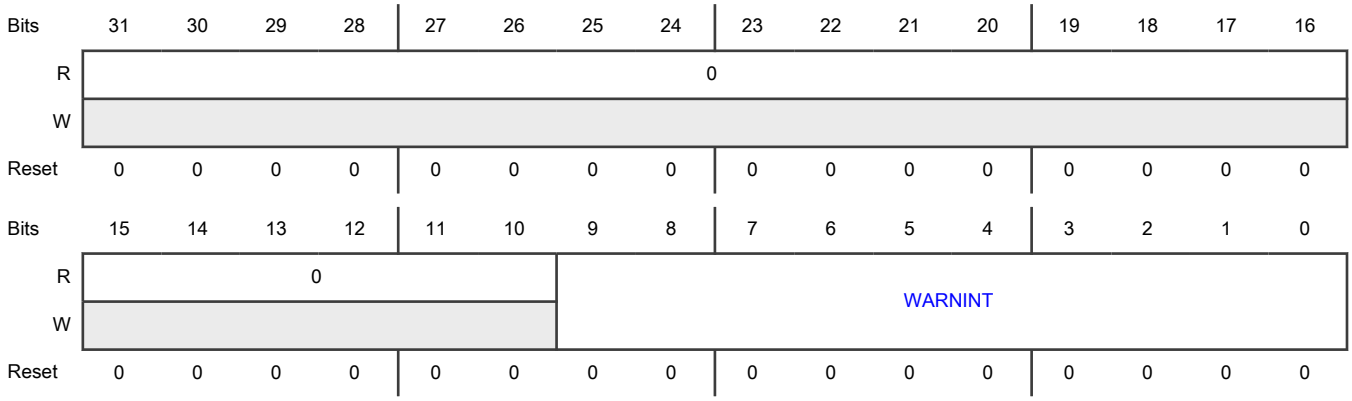
Register	Offset
WARNINT	14h

Function

Determines the value of TV[COUNT] that generates a watchdog interrupt. When this value is less than or equal to the value defined by WARNINT, an interrupt generates after the subsequent WDCLK.

A match of the WWDT counter to WARNINT occurs when the lower 10 bits of the timer counter have the same value as the lower 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1023 watchdog timer counts (4096 watchdog clocks) for the interrupt to occur before a watchdog event. If WARNINT is 0, then the interrupt occurs at the same time as the watchdog event.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 WARNINT	Watchdog Warning Interrupt Compare Value Specifies a timer value that generates a warning interrupt.

51.6.1.7 Window Compare Value (WINDOW)

Offset

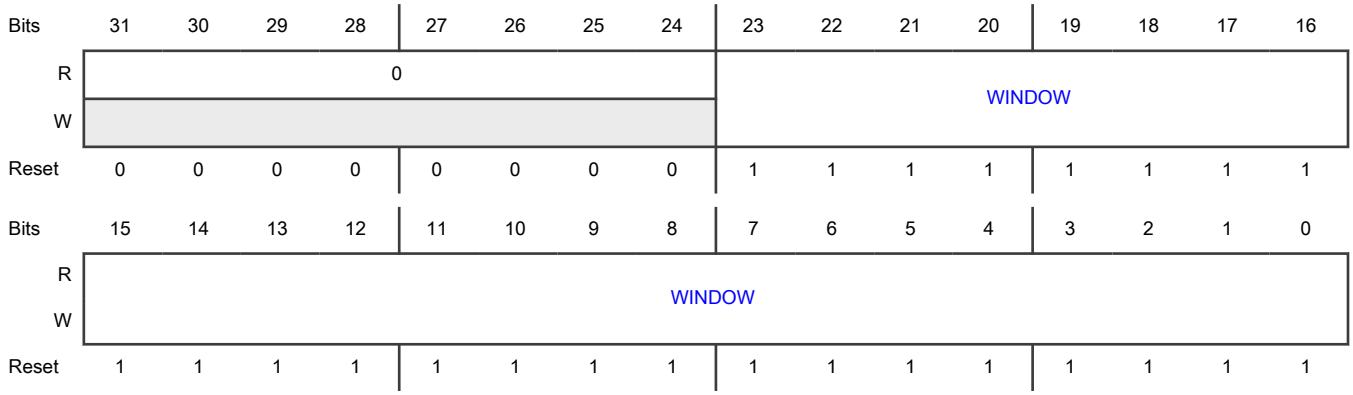
Register	Offset
WINDOW	18h

Function

Determines the highest timer value (TV[COUNT]) allowed during a watchdog feed. If a feed sequence occurs when the value of TV[COUNT] is greater than the value in WINDOW[WINDOW], a watchdog event occurs.

WINDOW[WINDOW] resets to the maximum value of TV[COUNT], where windowing is not in effect.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 WINDOW	Watchdog Window Value Specifies the highest timer value in which a watchdog feed can occur.

Chapter 52
Micro-Tick Timer (UTICK)

52.1 Chip-specific UTICK information

Table 441. Reference links to related information

Topic	Related module	Reference
Full description	UTICK	UTICK
Peripheral memory map		PBRG memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

52.1.1 Module instances

This device has one instance of the UTICK module, UTICK0.

52.1.2 UTICK clock

Enable the Watchdog oscillator that provides the Micro-Tick Timer clock via UTICK field in AHB Clock Control 1 AHBCLKCTRL1[UTICK] register.

See UTICK clocking.

52.2 Overview

UTICK is a 31-bit timer that counts down to 0 and then generates an interrupt. Thus, it provides a fixed time interval between interrupts. This is a simple, ultra-low-power timer that can run and wake up the chip from Low-power mode. See chip specific information for low power modes UTICK can wakeup.

52.2.1 Block diagram

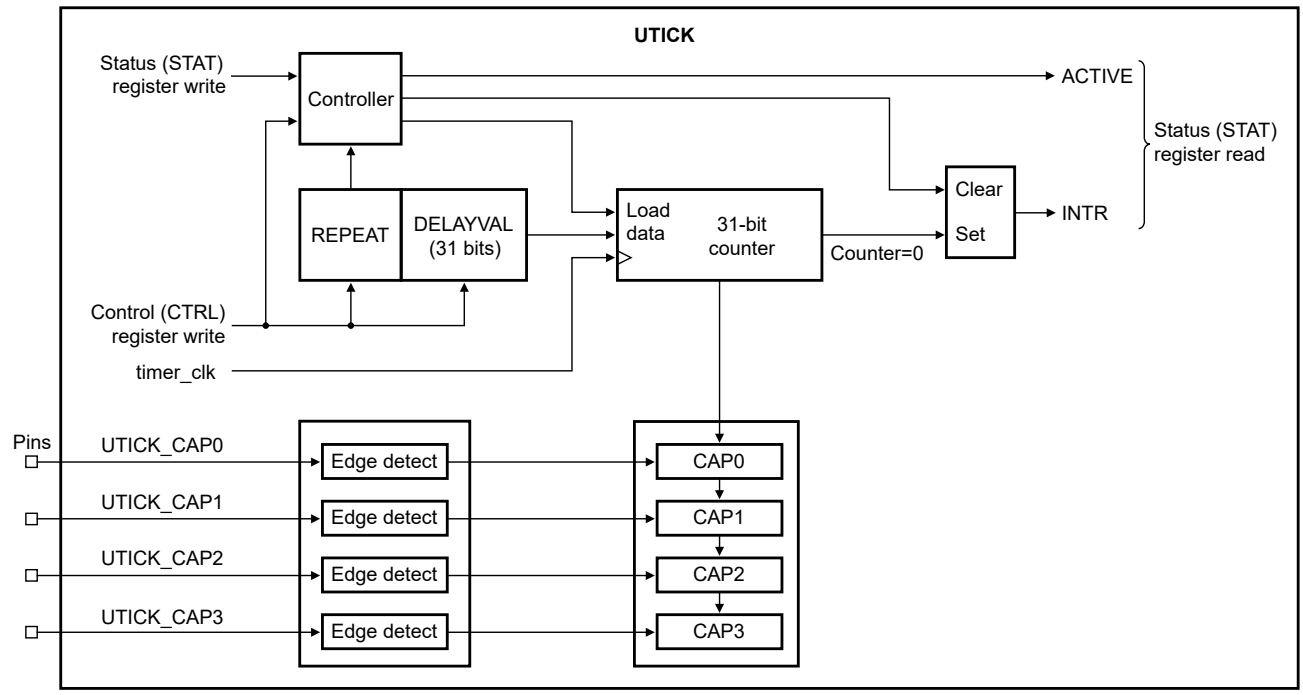


Figure 269. Block diagram

52.2.2 Features

- Wakes up the chip from Low-power mode.
- Starts with a nonzero value written to CTRL[DELAYVAL].
- Operates using interrupt or software polling mode.
- Includes Capture (CAP0 - CAP3) registers that use external pin transitions as a trigger.

52.3 Functional description

52.3.1 Operations

Perform the following procedure before using UTICK:

1. Configure CTRL[REPEAT] to either perform a one-time delay or continuous delay.
2. Configure CTRL[DELAYVAL] to specify the delay time value in terms of the timer clock.
3. Configure CFG[CAPEN*n*] to decide the number of events to capture (the maximum is four).
4. Configure CFG[CAPPOL*n*] to capture the polarity (whether to capture on a positive or negative edge of a signal).

52.3.2 Clocking

The UTICK clock is selectable from the following clock inputs:

Table 442. Clocking

Clock	Description
Global functional clock (timer_clk)	This clock is supposed to be on during normal operation. This clock is treated as asynchronous to pclk. **NOTE** See the Chip specific section for more details on UTICK clock.
System bus clock (pclk)	This clock is used only for register reads and writes.

52.3.3 Interrupts

The interrupt is generated when the tick_count reaches 1h so that on the next rising edge of timer_clk, it doesn't trigger again automatically. Its associated status field is STAT[INTR].

52.4 External signals

Signal	I/O	Description
UTICK_CAP0, UTICK_CAP1, UTICK_CAP2, UTICK_CAP3	I	Capture inputs. Configure the selected transition on a capture pin to load Capture (CAP0 - CAP3) with the counter value.

52.5 Initialization

To initialize UTICK:

1. Enable the clock to the UTICK register interface.
2. Enable UTICK interrupts for waking up from Low-power mode, configure Control (CTRL) register.
3. Configure Capture Configuration (CFG) register to enable UTICK capture functions and selects the polarity of capture triggers.
4. Enable the oscillator that provides the UTICK clock, after writing values corresponding registers UTICK timer works normally.

52.6 UTICK register descriptions

52.6.1 UTICK memory map

UTICK0 base address: 4001_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CTRL)	32	RW	0000_0000h
4h	Status (STAT)	32	RW	See section
8h	Capture Configuration (CFG)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
Ch	Capture Clear (CAPCLR)	32	W	See section
10h - 1Ch	Capture (CAP0 - CAP3)	32	R	0000_0000h

52.6.2 Control (CTRL)

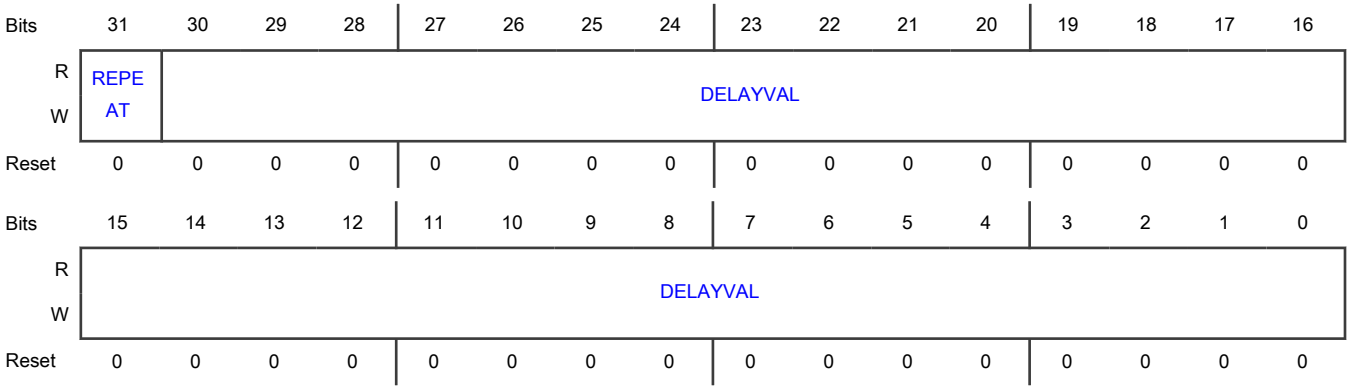
Offset

Register	Offset
CTRL	0h

Function

Resets the counter, which means a new interval is measured if one is in progress.

Diagram



Fields

Field	Function
31 REPEAT	Repeat Delay Specifies the frequency of repeat delay. 0b - One-time delay 1b - Delay repeats continuously
30-0 DELAYVAL	Tick Interval Specifies the tick interval. The resulting delay is (DELAYVAL + 1) periods of the timer clock. The minimum usable tick interval is one clock cycle, for a delay of two timer clocks. The timer will be stopped when the tick interval is equal to zero.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000_0000_0000_0000_0000_0000_0000_0000b - Reserved All other values - Clock cycles as defined in the description

52.6.3 Status (STAT)

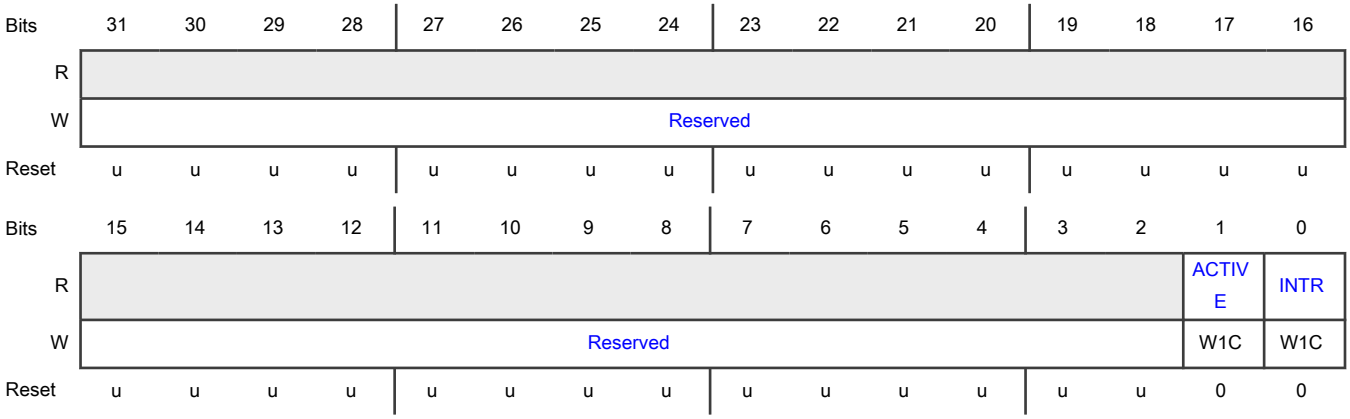
Offset

Register	Offset
STAT	4h

Function

Indicate the micro timer is working or stopping and any event on output compare channels. The bits are write one to clear.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 ACTIVE	Timer Active Flag Indicates whether UTICK is currently active. 0b - Inactive (stopped) 1b - Active
0 INTR	Interrupt Flag Indicates whether an interrupt is pending.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not pending 1b - Pending

52.6.4 Capture Configuration (CFG)

Offset

Register	Offset
CFG	8h

Function

Enables micro-tick capture functions and selects the polarity of capture triggers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					CAPPOL3	CAPPOL2	CAPPOL1	CAPPOL0					CAPEN3	CAPEN2	CAPEN1	CAPEN0
W	Reserved								Reserved							
Reset	u	u	u	u	0	0	0	0	u	u	u	u	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11 CAPPOL3	Capture Polarity 3 Specifies the capture edge. 0b - Positive 1b - Negative
10 CAPPOL2	Capture Polarity 2 Specifies the capture edge.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Positive 1b - Negative
9 CAPPOL1	Capture-Polarity 1 Specifies the capture edge. 0b - Positive 1b - Negative
8 CAPPOL0	Capture Polarity 0 Specifies the capture edge. 0b - Positive 1b - Negative
7-4 —	Reserved
3 CAPEN3	Enable Capture 3 0b - Disable 1b - Enable
2 CAPEN2	Enable Capture 2 0b - Disable 1b - Enable
1 CAPEN1	Enable Capture 1 0b - Disable 1b - Enable
0 CAPEN0	Enable Capture 0 0b - Disable 1b - Enable

52.6.5 Capture Clear (CAPCLR)

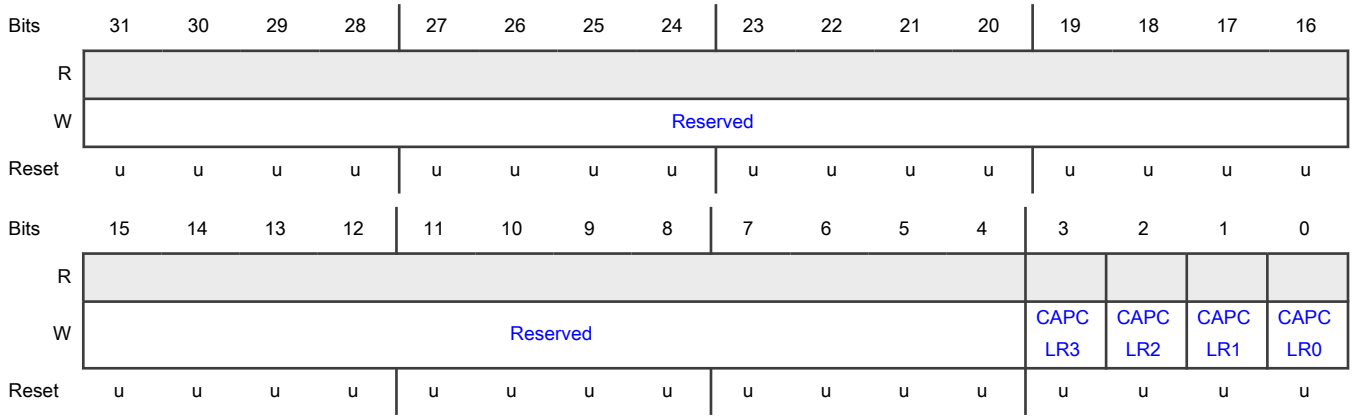
Offset

Register	Offset
CAPCLR	Ch

Function

Allows you to clear previous capture values, enabling new captures to take place.

Diagram



Fields

Field	Function
31-4 —	Reserved
3 CAPCLR3	Clear Capture 3 0b - Does nothing 1b - Clears the CAP3 register value
2 CAPCLR2	Clear Capture 2 0b - Does nothing 1b - Clears the CAP2 register value
1 CAPCLR1	Clear Capture 1 0b - Does nothing 1b - Clears the CAP1 register value
0 CAPCLR0	Clear Capture 0 0b - Does nothing 1b - Clears the CAP0 register value

52.6.6 Capture (CAP0 - CAP3)

Offset

Register	Offset
CAP0	10h
CAP1	14h
CAP2	18h

Table continues on the next page...

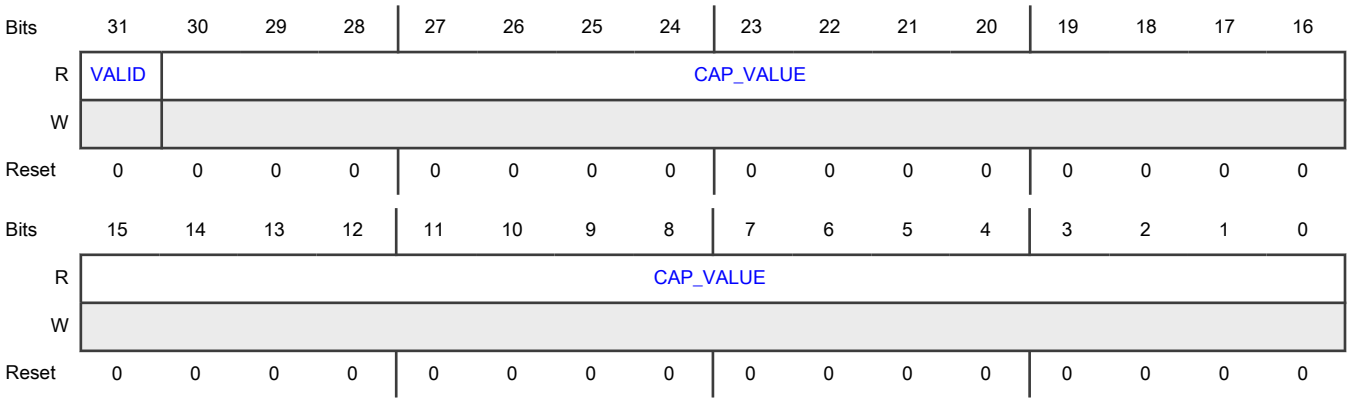
Table continued from the previous page...

Register	Offset
CAP3	1Ch

Function

Contains the micro-tick time value based on any previously captured events. Each capture register is associated with one of the capture trigger inputs.

Diagram



Fields

Field	Function
31 VALID	Captured Value Valid Flag Specifies whether a valid value is captured, based on a transition of the related UTICK_CAP*n* pin. Write 1 to the related field in Capture Clear (CAPCLR) to clear this flag. 0b - Valid value not captured 1b - Valid value captured
30-0 CAP_VALUE	Captured Value for the Related Capture Event Specifies the captured value for the related capture event. The captured value is 1 lower than the actual value of UTICK at the moment of the capture event.

Chapter 53
OS Event Timer (OSTIMER)

53.1 Chip-specific OSTIMER information

Table 443. Reference links to related information

Topic	Related module	Reference
Full description	OSTIMER	OSTIMER
Peripheral memory map		PBRG memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

53.1.1 Module instances

This device has one instance of the OSTIMER module, OSTIMER0.

53.2 Overview

OSTIMER includes a shared 42-bit timer and separate 42-bit match and capture functions for the Cortex-M33 core.

53.2.1 Block diagram

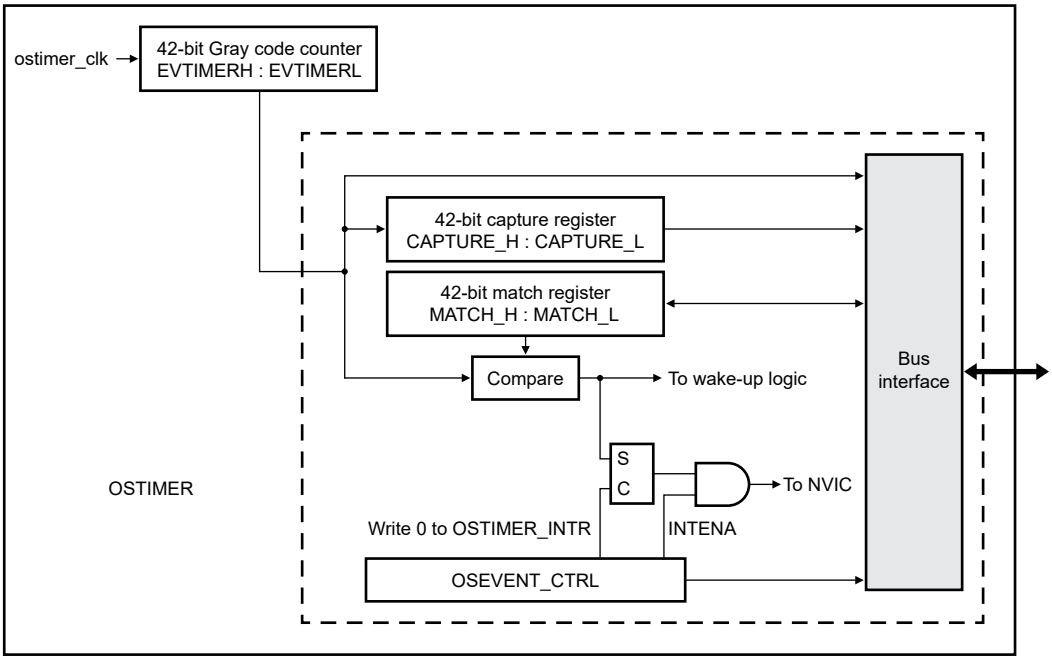


Figure 270. Block diagram

53.2.2 Features

- Provides a 42-bit Gray code counter (known as central EVTimer). Using Gray code means that the timer can run at a frequency unrelated to any CPU clock and can still be read by a CPU without a synchronization delay. Gray code is a reflected binary code that changes in a single-bit position for each increment.
- Supports separate functions for each CPU:
  - A capture register can copy the main counter value when triggered by a CPU request.
  - A match register can be compared to the main counter and can optionally generate an interrupt or wake-up event.

53.3 Functional description

The following sections describe the functional details of this module.

53.3.1 Shared event or timestamp timer

The 42-bit shared EVTimer initializes after chip power-up and then counts up continuously. The typical clock for this timer is the 1 MHz low-power oscillator.

This timer is implemented as a Gray code counter that enables the counter to be read asynchronously by any of the processing domains or captured by a capture register running on an asynchronous clock. The output of this shared EVTimer is connected to the processor-specific submodules.

53.3.2 CPU-specific match, capture, and interrupt generation

The submodules associated with each CPU include:

- A separate bus interface.
- 42 bits of capture values, available in CAPTURE_L and CAPTURE_H.
- 42 bits of match values, available in MATCH_H and MATCH_L.
- A control register, which includes an interrupt request flag and an interrupt enable field.

Capture registers	42 bits of capture values are available in CAPTURE_L and CAPTURE_H for each CPU. When the CPU issues a capture command, capture registers store the current value of the central EVTimer.
Match registers and interrupt request	42 bits of match values are available in MATCH_H and MATCH_L for each bus interface. The EVTimer output is compared against this combined value for interrupt or wake-up generation. A match to this register pair sets OSEVENT_CTRL[OSTIMER_INTRFLAG], which you can enable to generate an interrupt or wake-up request. You must write values to the match registers in Gray code. OSEVENT_CTRL[MATCH_WR_RDY] must be 0 before you write to the match registers. When you write a new value to the match registers, you must write to MATCH_L before the write to MATCH_H.

53.3.3 Clocking

The OSTIMER uses three clock signals: hclk, capture_clk, and ostimer_clk.

1. Hclk is the IPS interface clock for both the Arm and DSP sub-module, being used in all the registers and write/read operations.
2. Ostimer_clk is the counter clock, being used in the timer itself and in the intr_wakeup signals. It is expected to be running in a low frequency (~1MHz).
3. capture_clk is the Arm and DSP clock, it is used to register the capture signal “capture_load”.

53.3.4 Reset

For OSTIMER, the reset function is more complicated as compared to most other functions.

- Only the POR and software reset can reset the shared EVTimer. "Software reset" refers to a complete reset of the module.
- The capture and match registers are reset by a system reset. "System reset" refers to a reset from the RESET*n* pin, SYSRESETREQ from the CPU, a POR, or any other internal low-voltage resets.
- The control register (OSEVENT_CTL) is reset by the system reset or by software reset. However, the MATCH_WR_RDY bit can only be reset by system reset.

The following table shows a list of resets for OSTIMER registers.

Table 444. OSTIMER registers and their corresponding resets

Registers	POR	Software reset
EVTIMERL and EVTIMERH	Yes	Yes
CAPTURE_L and CAPTURE_H	Yes	No
OSEVENT_CTRL	Yes	Yes

53.3.5 Interrupts

If the interrupt request is enabled (i.e., OSEVENT_CTRL[OSTIMER_INTENA]= 1), the OSTIMER will generate interrupt when the central 42-bit EVTimer and the value programmed in MATCH_L and MATCH_H matches.

No interrupt generates if the interrupt request is disabled (i.e., OSEVENT_CTRL[OSTIMER_INTENA]= 0).

53.4 External signals

This module has no external signals.

53.5 Initialization

Perform this procedure to initialize OSTIMER:

1. Enable the OSTIMER clock. This enables the register interface and the peripheral function clock.
2. Select a clock source for OSTIMER.
3. Clear the OSTIMER peripheral reset.
4. Enable the interrupt that OSTIMER provides to the interrupt controller (set OSEVENT_CTRL[OSTIMER_INTRFLAG]= 1). This allows the chip to wake up from Deep Sleep mode.

53.6 Memory map and register definition

This section includes the module's memory map and detailed descriptions of all registers.

53.6.1 OSTIMER register descriptions

53.6.1.1 OSTIMER memory map

OSTIMER0 base address: 4004_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	EVTIMER Low (EVTIMERL)	32	R	0000_0000h
4h	EVTIMER High (EVTIMERH)	32	R	0000_0000h
8h	Local Capture Low for CPU (CAPTURE_L)	32	R	0000_0000h
Ch	Local Capture High for CPU (CAPTURE_H)	32	R	0000_0000h
10h	Local Match Low for CPU (MATCH_L)	32	RW	FFFF_FFFFh
14h	Local Match High for CPU (MATCH_H)	32	RW	FFFF_FFFFh
1Ch	OSTIMER Control for CPU (OSEVENT_CTRL)	32	RW	0000_0000h

53.6.1.2 EVTIMER Low (EVTIMERL)

Offset

Register	Offset
EVTIMERL	0h

Function

Contains the immediate value of the central EVTimer. This is a Gray encoded counter register that is not synchronized with the bus clock.

Because two 32-bit reads are required to retrieve the entire counter value, it is possible for the counter to roll over between the reads of EVTIMERL and EVTIMERH. Gray encoding ensures that even if this roll over occurs, the value read represents either the counter value immediately preceding or immediately following the rollover, if the bus clock rate is at least twice the module's clock rate (nominally 1 MHz), or if the bus is clocked by the same 1 MHz LP oscillator that provides the clock to the module.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EVTIMER_COUNT_VALUE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVTIMER_COUNT_VALUE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 EVTIMER_COUNT_VALUE	EVTimer Count Value Specifies the current value of the lower 32 bits of the central 42-bit EVTimer when you read this register. **NOTE** Only one physical EVTimer exists and is readable from all domains.

53.6.1.3 EVTIMER High (EVTIMERH)

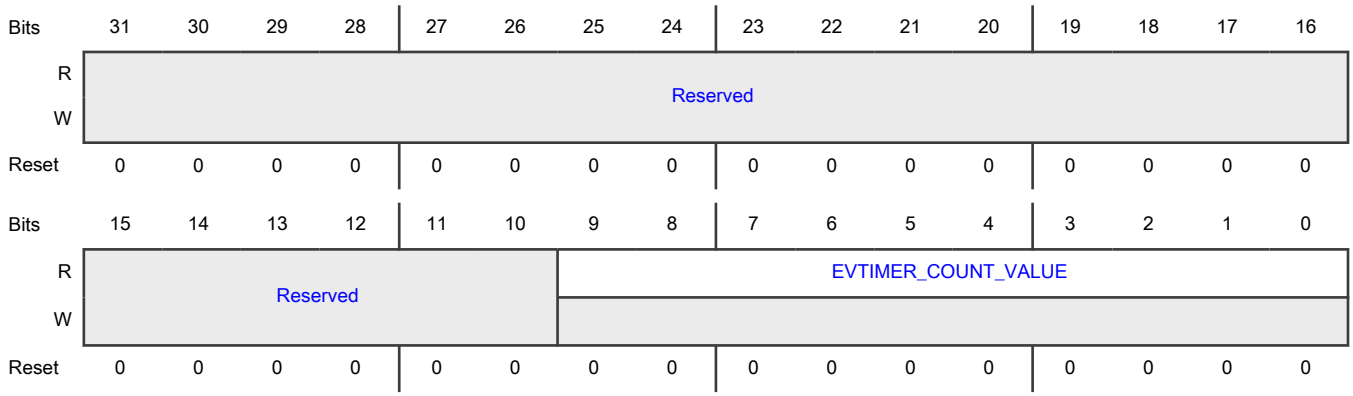
Offset

Register	Offset
EVTIMERH	4h

Function

Contains the higher value of the central EVTimer. This register resets only after POR or a software reset.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 EVTIMER_COUNT_VALUE	EVTimer Count Value Specifies the current value of the upper 10 bits of the central 42-bit EVTimer when you read this register. **NOTE** Only one physical EVTimer exists and is readable from all domains.

53.6.1.4 Local Capture Low for CPU (CAPTURE_L)

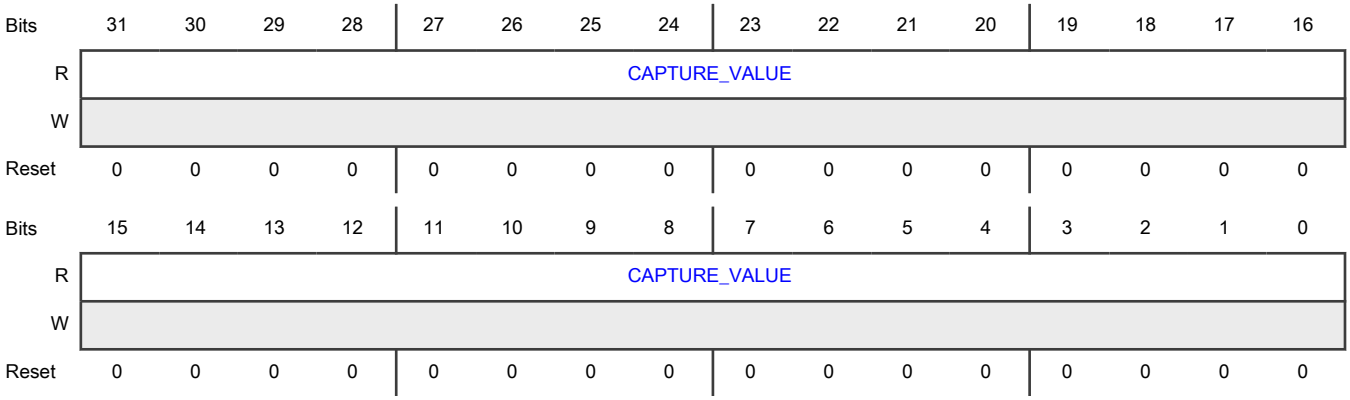
Offset

Register	Offset
CAPTURE_L	8h

Function

Contains the value of the central EVTimer when captured by the CPU. This register is Gray encoded to match EVTimer. You must not execute a new capture command between reading CAPTURE_L and CAPTURE_H, to avoid retrieving incoherent results.

Diagram



Fields

Field	Function
31-0 CAPTURE_VALUE	EVTimer Capture Value Specifies the value of the lower 32 bits of the central 42-bit EVTimer when you read this register, at the time that the CPU generated the last capture signal. A separate pair of CAPTURE_L and CAPTURE_H is implemented for each CPU. Each CPU reads its own capture value at the same pair of addresses.

53.6.1.5 Local Capture High for CPU (CAPTURE_H)

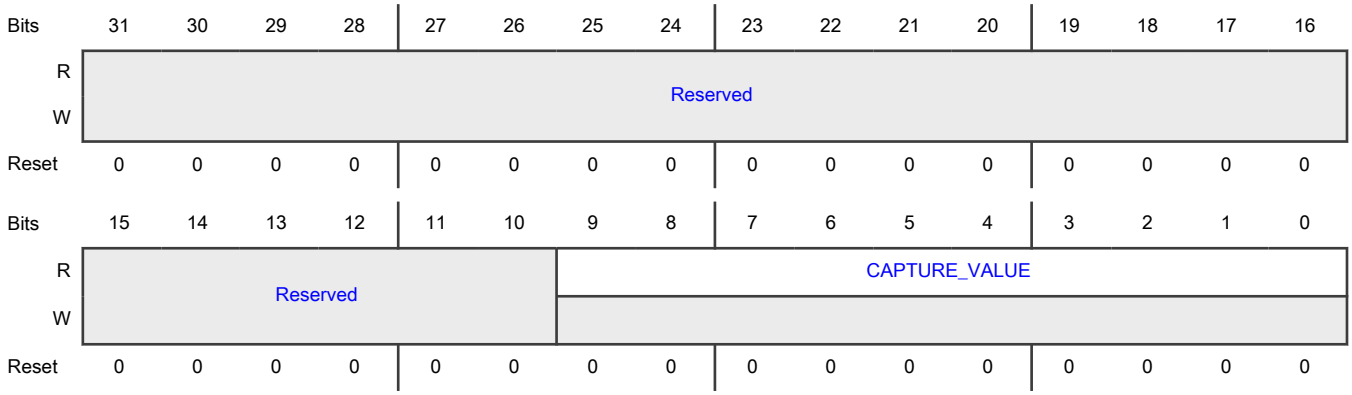
Offset

Register	Offset
CAPTURE_H	Ch

Function

Contains the value of the central EVTimer when captured by the CPU. This register is Gray encoded to match EVTimer. You must not execute a new capture command between reading CAPTURE_L and CAPTURE_H, to avoid retrieving incoherent results.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 CAPTURE_VALUE	EVTimer Capture Value Specifies the value of the upper 10 bits of the central 42-bit EVTimer when you read this register, at the time the CPU generated the last capture signal (using CMSIS C function "__SEV();").

53.6.1.6 Local Match Low for CPU (MATCH_L)

Offset

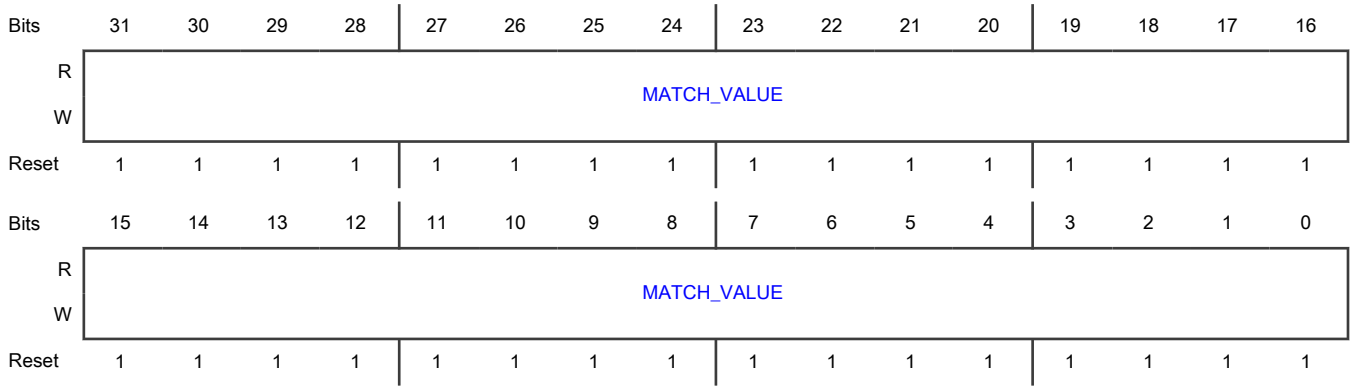
Register	Offset
MATCH_L	10h

Function

Compares the match value with the central EVTimer value. This register is Gray encoded to match EVTimer. The value in this pair of registers is stable, so there is no need for multiple reads. When writing to the match register pair, you must write to MATCH_L first, followed by the write to MATCH_H.

You must not initiate a second write to the match register pair until the first write is complete (which is a minimum of three bus clocks followed by a write to MATCH_H). OSEVENT_CTRL[MATCH_WR_RDY] indicates when it is safe to reload the match registers. You do not need to read OSEVENT_CTRL[MATCH_WR_RDY] if an interrupt has already occurred because of the previous match value, or the required time period has elapsed.

Diagram



Fields

Field	Function
31-0 MATCH_VALUE	EVTimer Match Value Compares the value (lower 32 bits) of the MATCH_L and MATCH_H register pair with the central EVTimer value. If both the values match, the CPU generates an interrupt request if the interrupt is enabled. A separate pair of MATCH_L and MATCH_H is implemented for each CPU reading its own local value at the same pair of addresses.

53.6.1.7 Local Match High for CPU (MATCH_H)

Offset

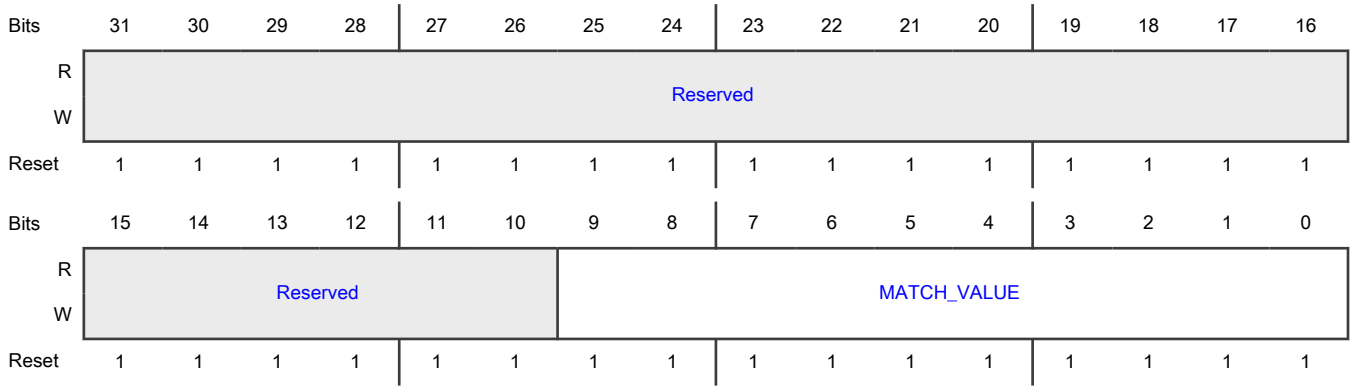
Register	Offset
MATCH_H	14h

Function

Compares the match value with the central EVTimer value. This register is Gray encoded to match EVTimer. The value in this pair of registers is stable, so there is no need for multiple reads. When writing to the match register pair, you must write to MATCH_L first, followed by the write to MATCH_H.

You must not initiate a second write to the match register pair until the first write is complete (which is a minimum of three bus clocks followed by a write to MATCH_H). OSEVENT_CTRL[MATCH_WR_RDY] indicates when it is safe to reload the match registers. You do not need to read OSEVENT_CTRL[MATCH_WR_RDY] if an interrupt has already occurred because of the previous match value, or the required time period has elapsed.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 MATCH_VALUE	EVTimer Match Value Compares the value (upper 10 bits) of the MATCH_L and MATCH_H register pair with the central EVTimer value. If both the values match, the CPU generates an interrupt request if the interrupt is enabled.

53.6.1.8 OSTIMER Control for CPU (OSEVENT_CTRL)

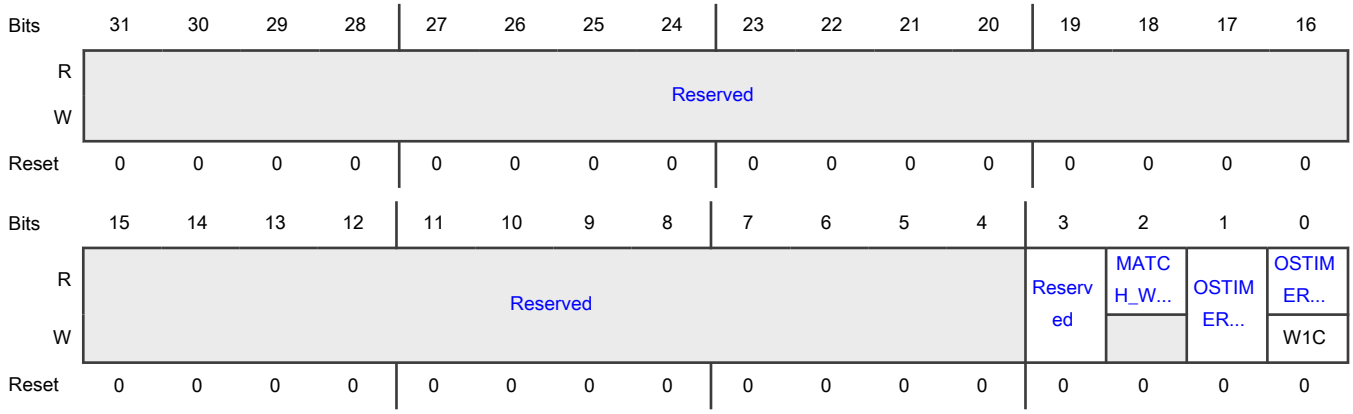
Offset

Register	Offset
OSEVENT_CTRL	1Ch

Function

Provides the interrupt flag and interrupt enable signals for each CPU. A separate OSEVENT_CTRL register is implemented for each CPU. Each CPU reads its own local value at the same address.

Diagram



Fields

Field	Function
31-4 —	Reserved
3 —	Reserved
2 MATCH_WR_RDY	EVTimer Match Write Ready Specifies that when this field is low, it is safe to reload (write) to the match registers. In typical applications, it is not necessary to read this field. The 42-bit match register value is transferred from a pair of shadow registers to the active match registers after you write to MATCH_H. You must not initiate a second write to MATCH_L and MATCH_H until after this transfer completes. This field becomes 0 after the transfer completes. It is not necessary to read this status field, if an interrupt has already occurred because of the first match value, or if it is certain (via some other means) that the required period of time (3 bus clocks) has elapsed.
1 OSTIMER_INTENA	Interrupt or Wake-Up Request This bit is to enable/disable interrupt request. If the value is 1, then this field asserts an interrupt or wake-up request to the domain processor. Otherwise, the interrupt or wake-up requests are blocked. 0b - Interrupts blocked 1b - Interrupts enabled
0 OSTIMER_INTRFLAG	Interrupt Flag Sets an interrupt flag when a match occurs between the central 42-bit EVTimer and the value programmed in the MATCH_L and MATCH_H register pair for the associated CPU. Writes to clear this field are asynchronous. You must write 1 to this field before writing a new match value into MATCH_L and MATCH_H.

Chapter 54
Enhanced Flex Pulse Width Modulator (PWM)

54.1 Chip-specific PWM information

Table 445. Reference links to related information

Topic	Related module	Reference
Full description	PWM	PWM
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See FlexPWMn_SMn_EXTAn registers in INPUTMUX

NOTE

PWM uses system_clk.

54.1.1 Module instances

This device has two instances of the PWM module, PWM0 and PWM1. Neither instance of the module supports the NanoEdge placement feature. Fine edge control is simulated by a dithering process.

54.1.2 PWM pins and signals information

In this device, PWMn_EXTB input is not used; only PWMn_EXTA input is used.

54.1.3 Reset value of Fault Status Register (FSTS0)

The reset value of FST0 register on this device is 0x0F0F.

54.2 Overview

The Pulse Width Modulator (PWM) module contains PWM submodules, each of which can be used to control a single half-bridge power stage. Fault channel support is provided.

This module generates various switching patterns, including highly sophisticated waveforms. It is ideal for controlling different Switched Mode Power Supplies (SMPS) topologies.

54.2.1 Block diagram

The following figure shows the PWM block diagram.

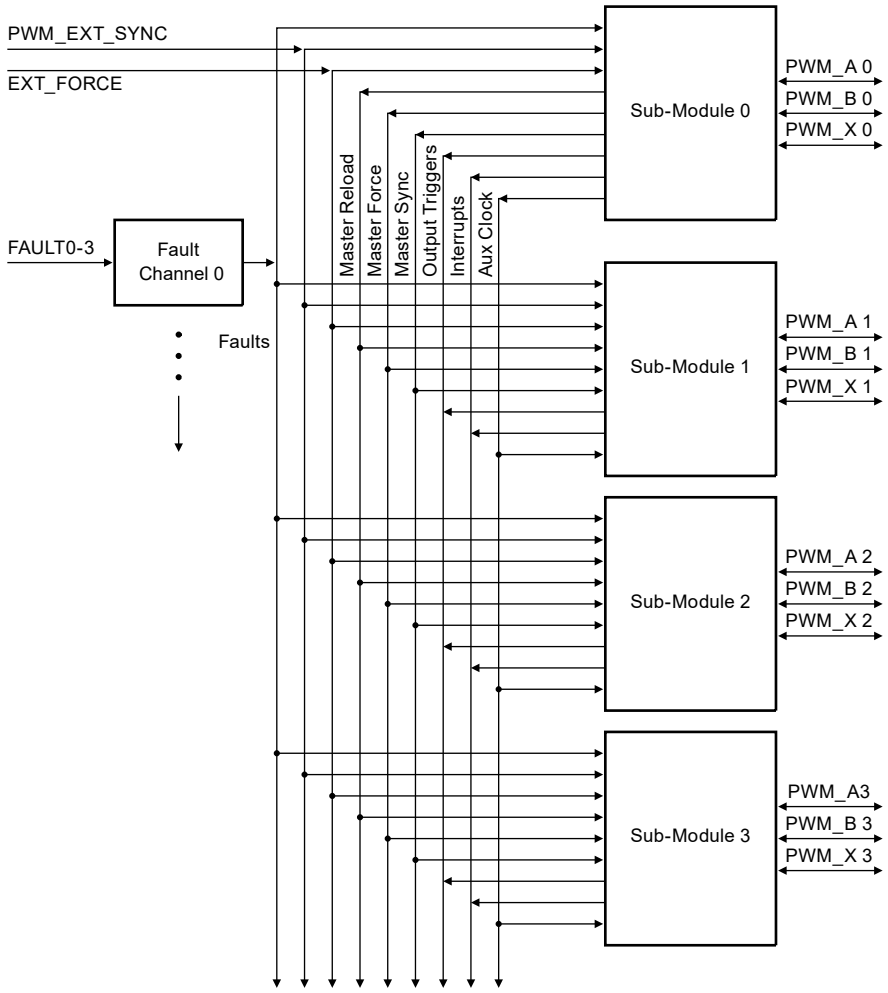


Figure 271. PWM block diagram

54.2.2 Features

Following are the features of PWM:

- 16-bit resolution for center, edge-aligned, and asymmetrical PWMs
- Dithering to simulate enhanced resolution when fine edge placement is not available
- PWM outputs that can operate as complementary pairs or independent channels
- Ability to accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Support for synchronization to external hardware or other PWM
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs

- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output
- All outputs can be programmed to change simultaneously via a FORCE_OUT event
- PWM_X pin can optionally output a third PWM signal from each submodule
- Channels not used for PWM generation can be used for buffered output compare functions and for input capture functions
- Enhanced dual edge capture functionality

54.3 Functional description

54.3.1 PWM submodule

The following figure shows the PWM Submodule Block Diagram.

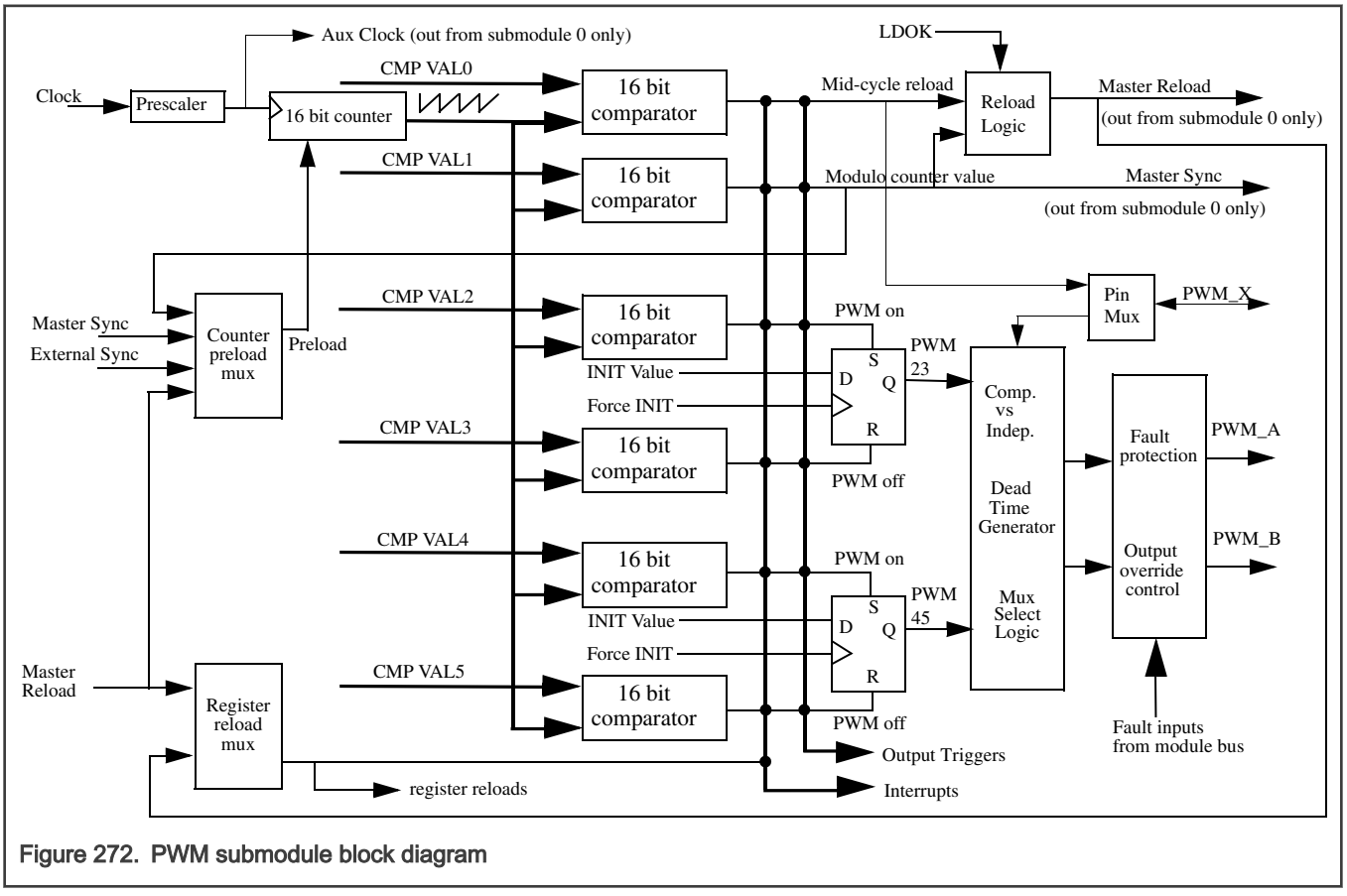


Figure 272. PWM submodule block diagram

54.3.2 PWM capabilities

This section describes some capabilities of the PWM module.

54.3.2.1 Center aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in Figure 273.

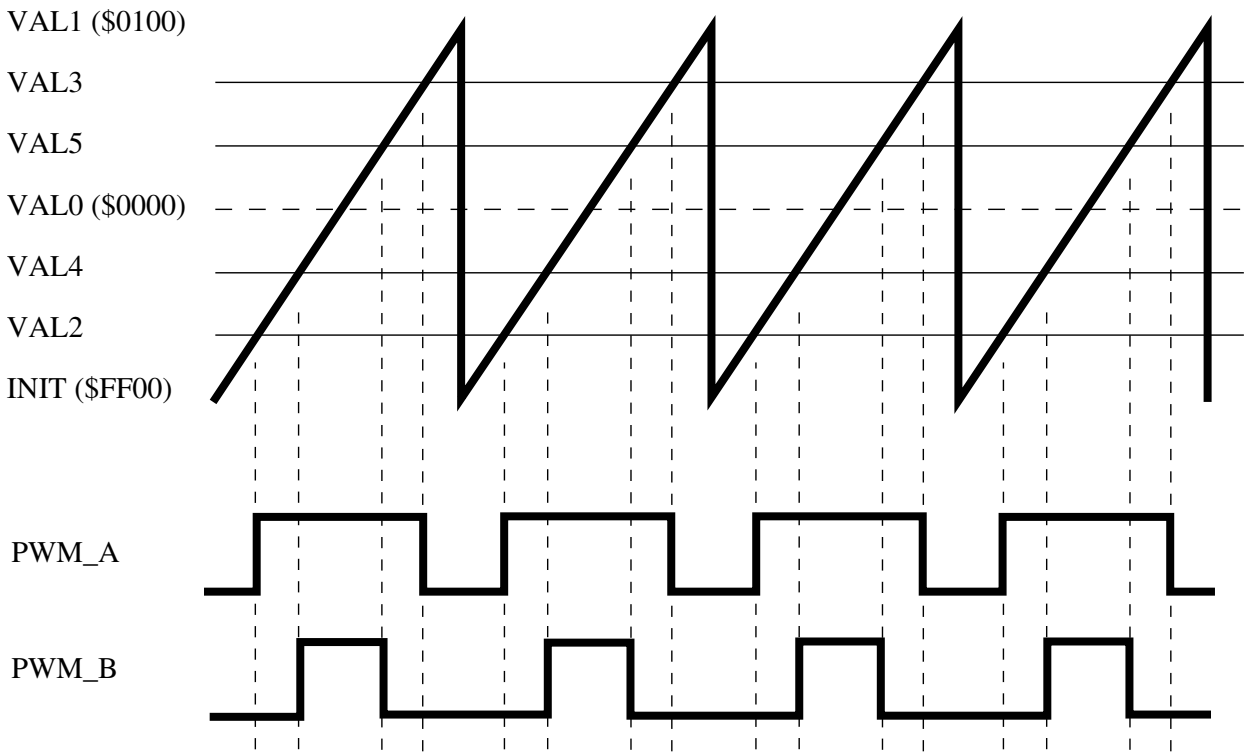


Figure 273. Center aligned PWM example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn-on edge and the turn-off edge. This double-action edge generation provides the user control over the pulse width and also the relative alignment of the signal. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn-on and turn-off edge values.

Figure 273 also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user-specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means if each PWM's turn-on and turn-off edge values are same in numbers but different in their sign, the "on" portion of the output signal is centered around a count value of zero. Therefore, only one PWM value is calculated in software and then this value and its negative are provided to the submodule as the turn-off and turn-on edges respectively. This technique results in a pulse width consists of an odd number of timer counts. If all PWM signal edge calculations follow this convention, then the signals will be center aligned with each other, which is the goal. The center alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

54.3.2.2 Edge aligned PWMs

Figure 274 shows the results of edge aligned operation when the turn-on edge for each pulse is specified to be the INIT value. Therefore, only the turn-off edge value needs to be periodically updated to change the pulse width.

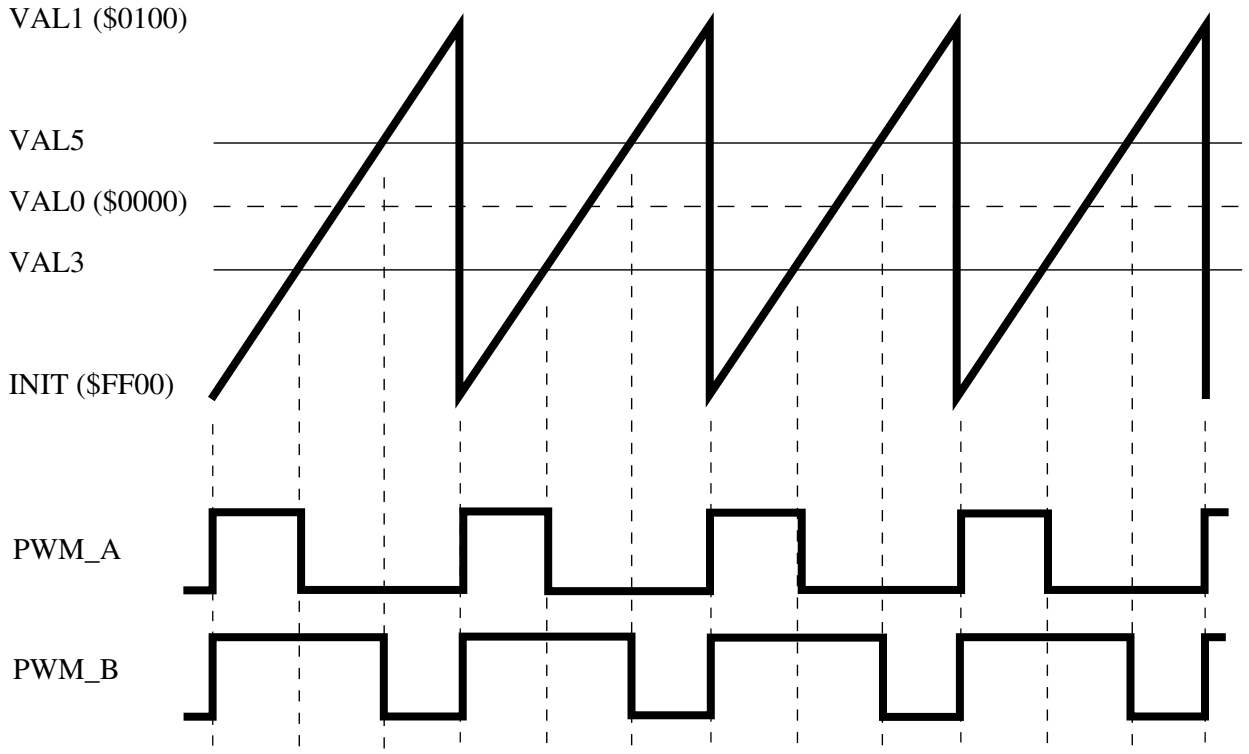


Figure 274. Edge aligned example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. Use "bipolar" PWMs to drive an H-bridge, where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% generate negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn-off edge value and the motor inverter voltage, including the sign. Therefore, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

54.3.2.3 Phase shifted PWMs

In the previous sections, the benefits of the signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases are applied to the turn-on and turn-off edges of different PWM signals, the signals will be phase shifted to each other, as shown in Figure 275. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does not affect the duty cycle so average load voltage is not affected.

If the outputs of submodules 1 - 3 need to be delayed from the output of submodule 0 (and from each other), instead of just creating a phase delay by adding an offset to the turn on and turn off times of the different submodules, another method is to use the PHASEDLY registers for submodules 1 - 3 to indicate their delay from the submodule 0 timing. This method can be used when the master sync signal from submodule 0 is selected as the initialization source (CTRL2[INIT_SEL]==b10). This method allows all of the submodules to be programmed with the same turn on and turn off time but submodules 1 - 3 can still be delayed the time from submodule 0.

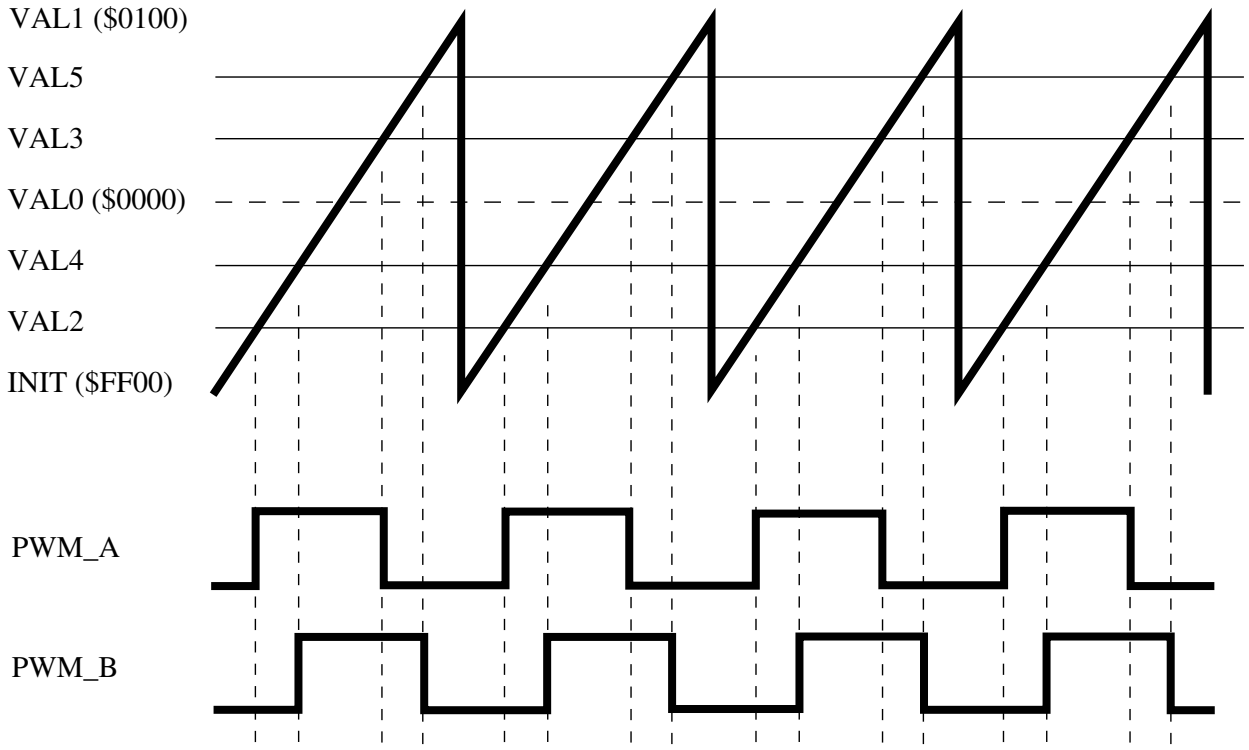


Figure 275. Phase shifted outputs - example for 1 submodule

An additional benefit of phase shifted PWMs is shown in Figure 276. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to generate a square wave with 50% duty cycle. This works for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% suitable for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

NOTE

The square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals.

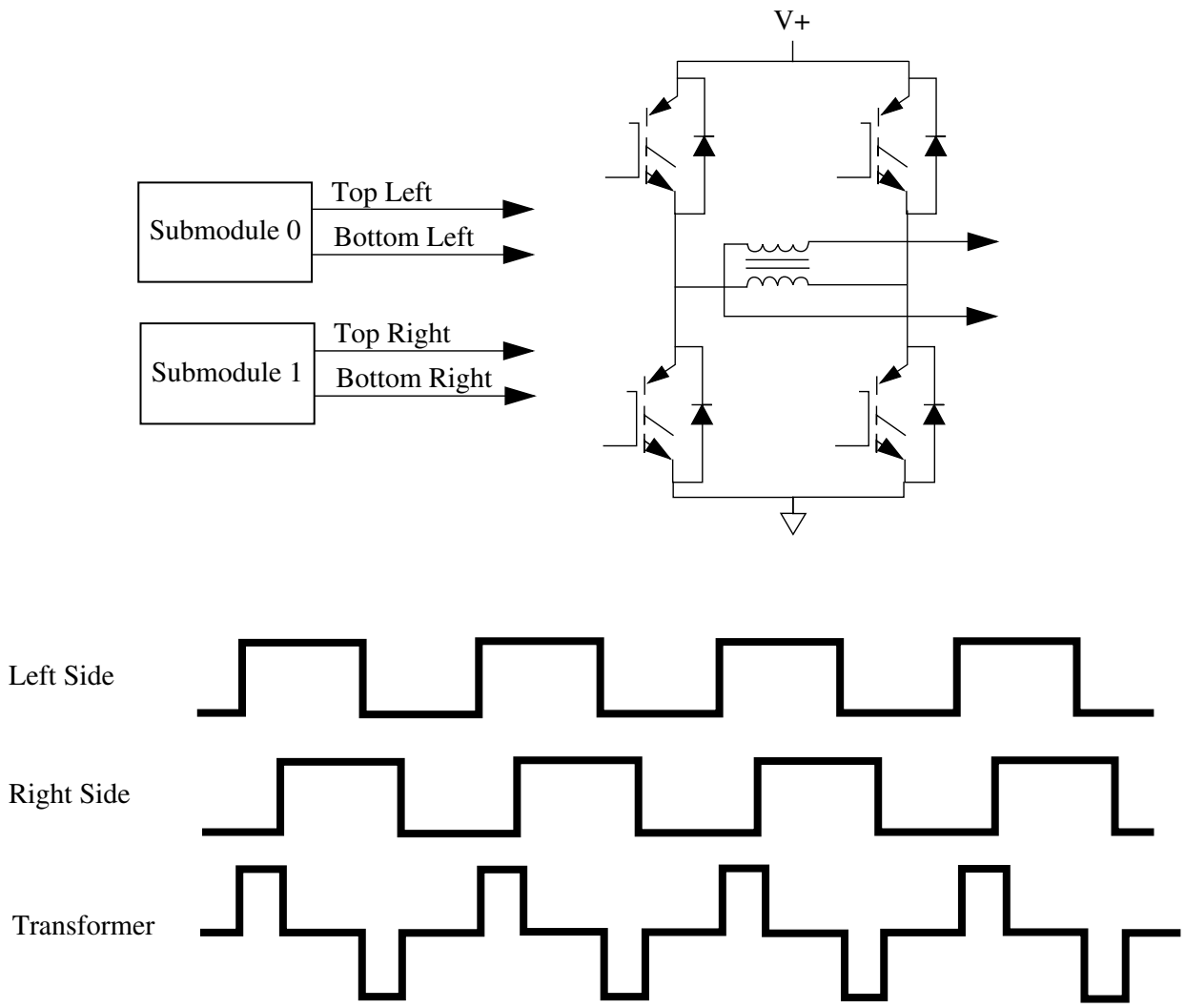


Figure 276. Phase Shifted PWMs Applied to a Transformer Primary

54.3.2.4 Double switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three-phase reconstruction. This method supports two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labeled as PWM_A in Figure 277) while VAL4 and VAL5 are used to generate the odd channel. The two channels (PWM23 or PWM_A and PWM45 or PWM_B from force out logic) are combined using XOR logic (force out logic) as the following figure shows. The DBLPWM signal can be run through the deadtime insertion logic.

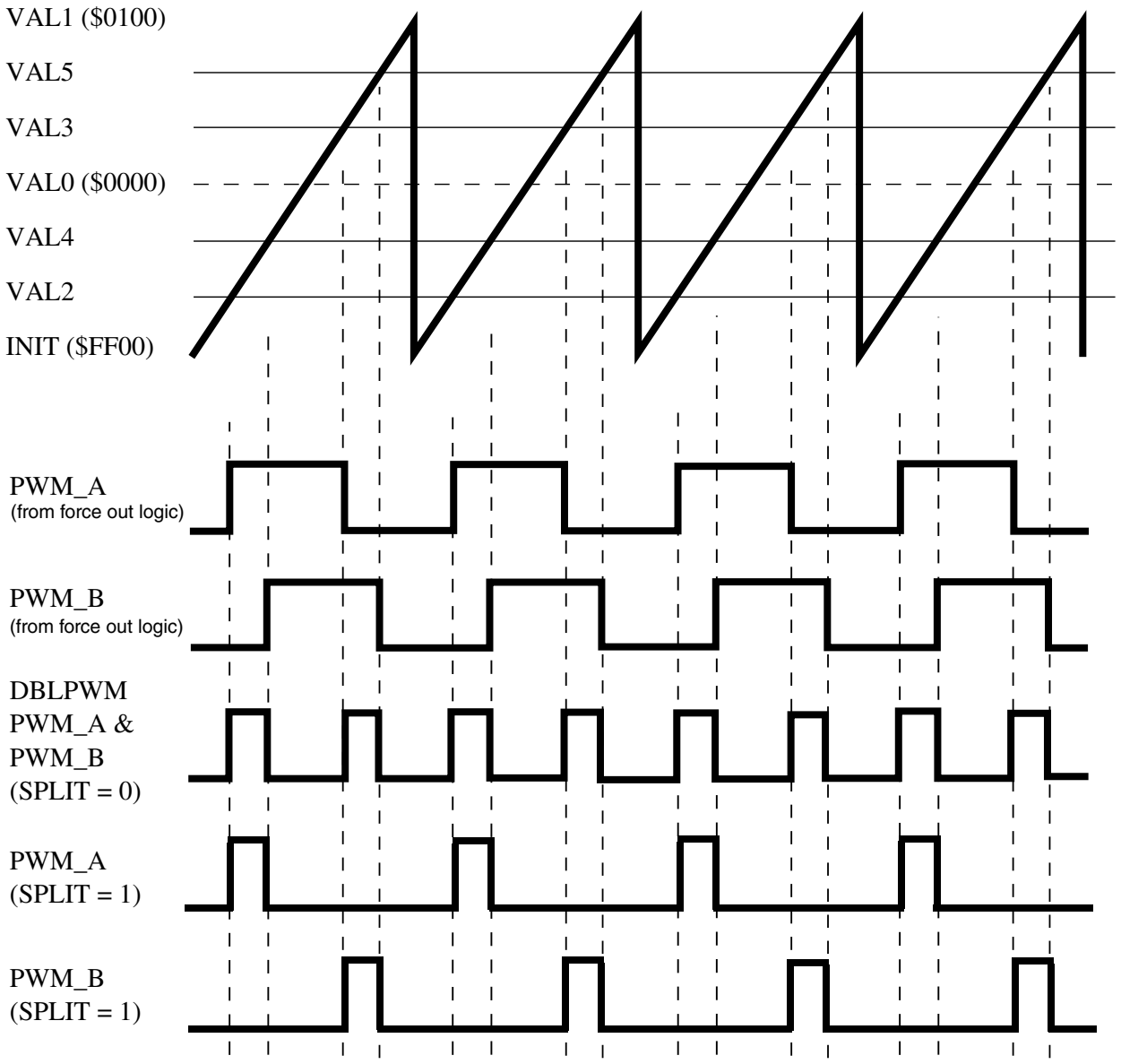


Figure 277. Double switching output example

54.3.2.5 ADC triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. Figure 278 shows how this is accomplished. When specifying a complementary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means the other comparators are free to perform other functions. In this example, the software does not need to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

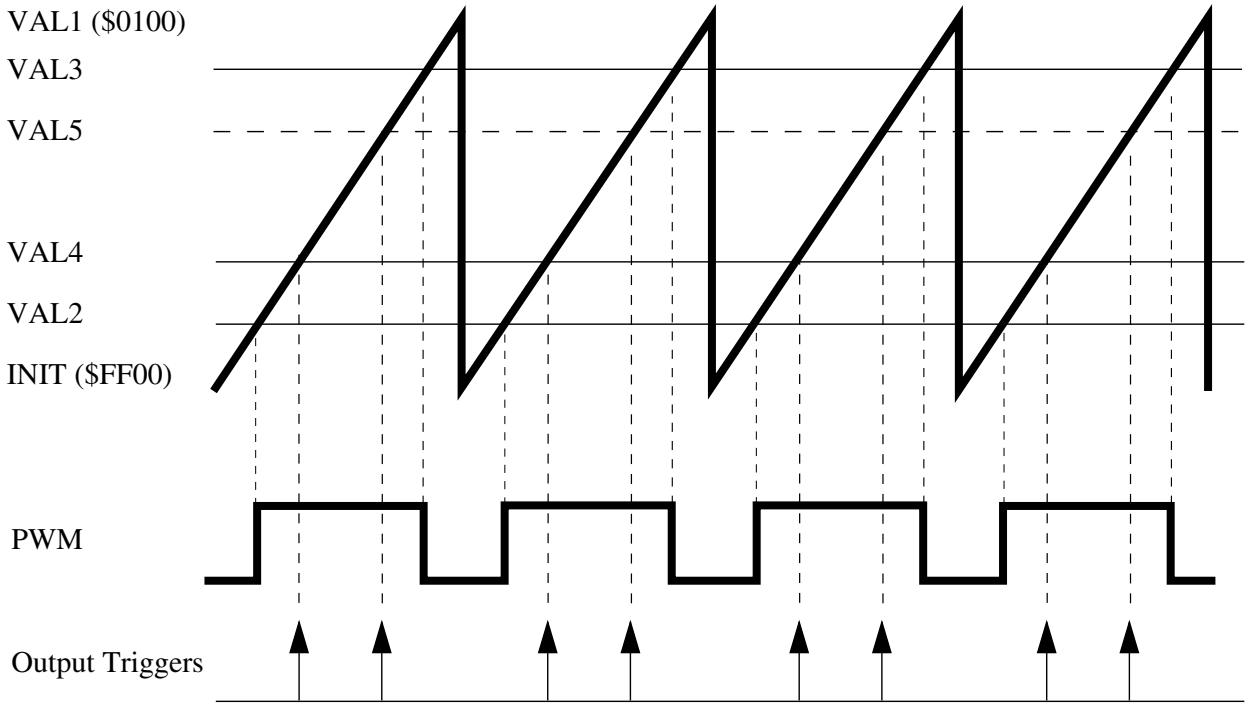


Figure 278. Multiple output trigger generation in hardware

Because each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. Figure 279 shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. You can use the lower-frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In Figure 279, *all* submodule comparators are shown being used for ADC trigger generation.

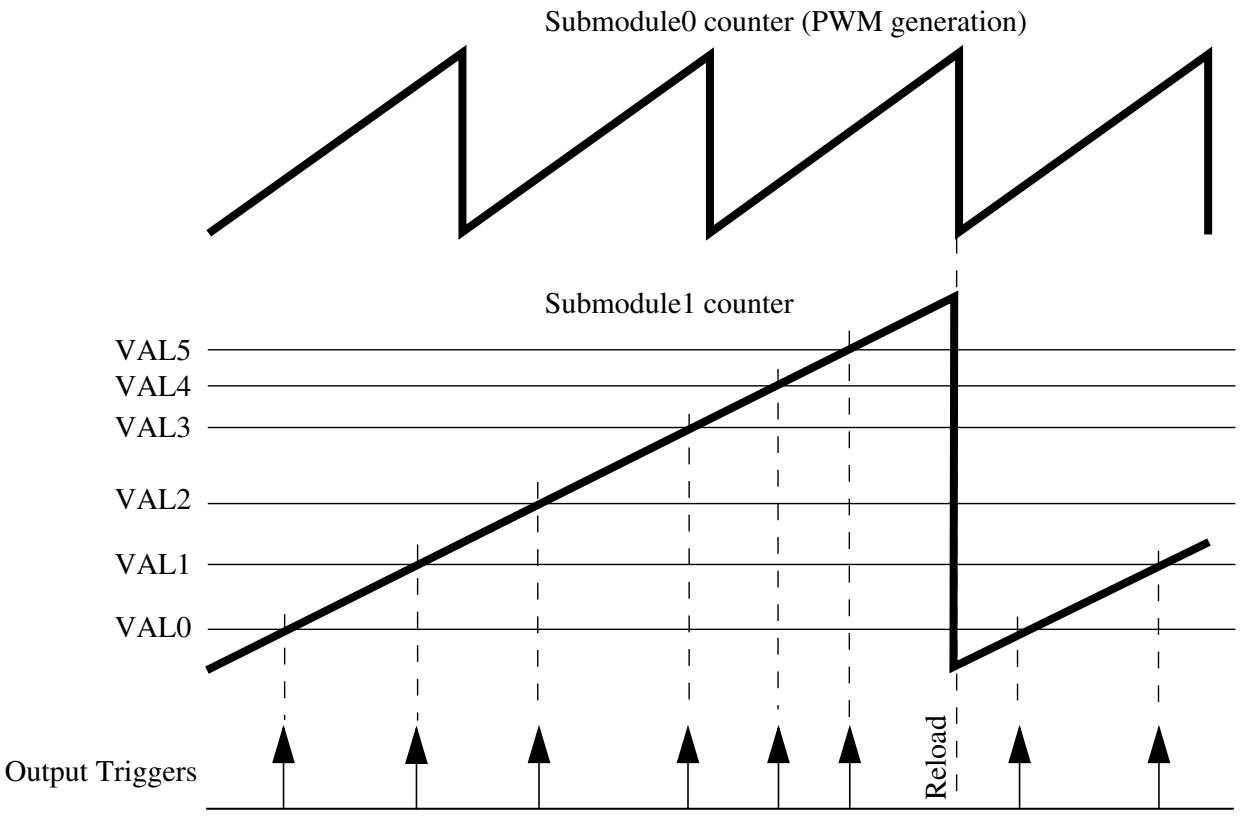


Figure 279. Multiple output triggers over several PWM cycles

54.3.2.6 Enhanced capture capabilities (E-Capture)

When a PWM pin is not used for PWM generation, it can be used to perform input captures. For PWM generation, both edges of the PWM signals are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. Programming the desired edge of each capture circuit, period, and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8-bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature counts a specified number of edge events and then performs a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

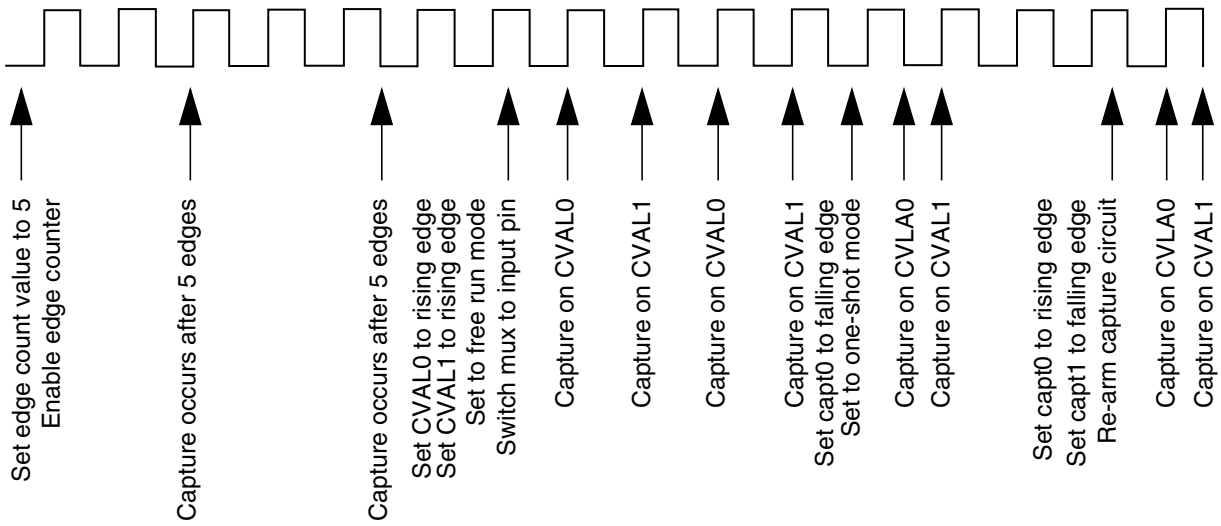


Figure 280. Capture capabilities of the E-Capture circuit

When a submodule is used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (for example, PWM_X) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16-bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is suited for the application. As shown in Figure 281, the output of a PWM power stage is connected to the PWM_X pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This results in new load pulse width data acquired every PWM cycle. To calculate the pulse width, subtract the CVAL0 register value from the CVAL1 register value. This measurement is beneficial when performing deadtime distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays. For details, refer to the separate discussion of deadtime distortion correction.

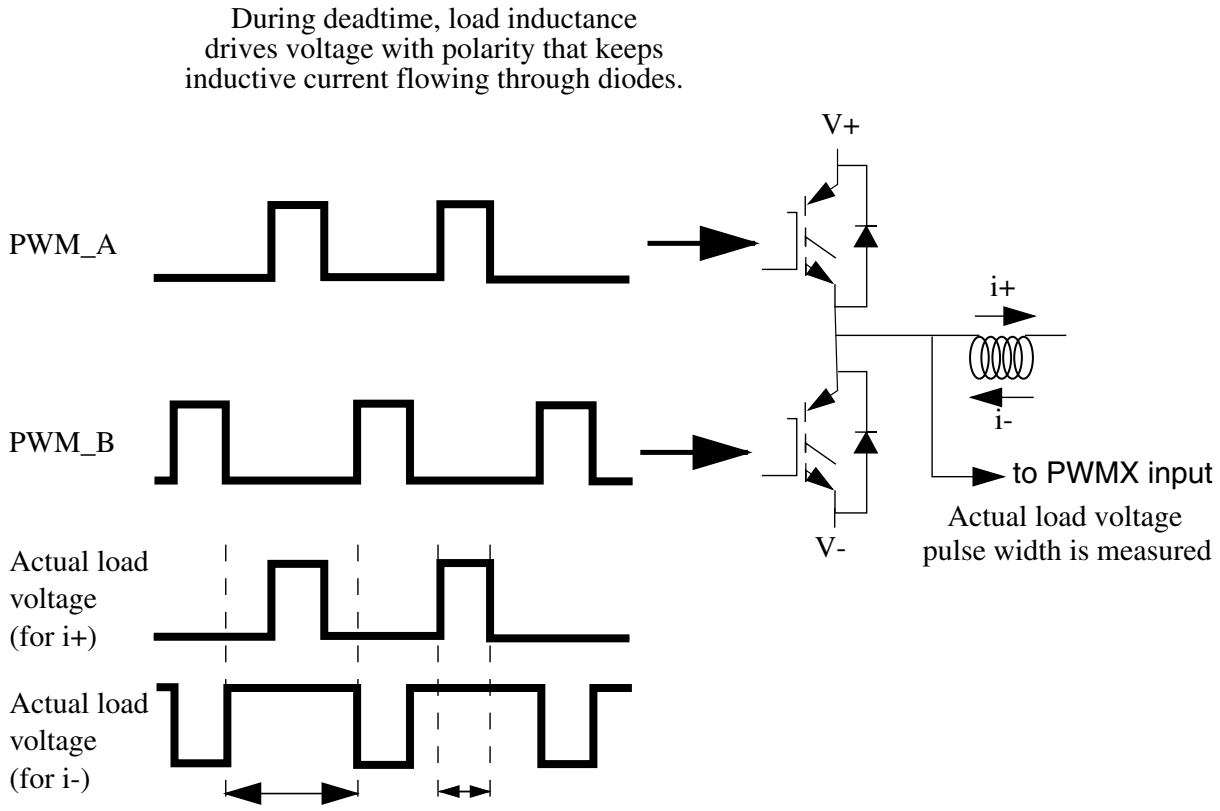


Figure 281. Output pulse width measurement possible with the E-capture circuit

54.3.2.7 Synchronous switching of multiple outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature is useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. All the changes occur immediately after the trigger event occurs eliminating any interrupt latency and also the changes occurs synchronously on all submodule outputs.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module, and in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that occur do not violate deadtime on the power stage when in complementary mode.

Figure 282 shows an application that can benefit from this feature. On a brushless DC motor in many cases, it is required to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of Figure 282 represent these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

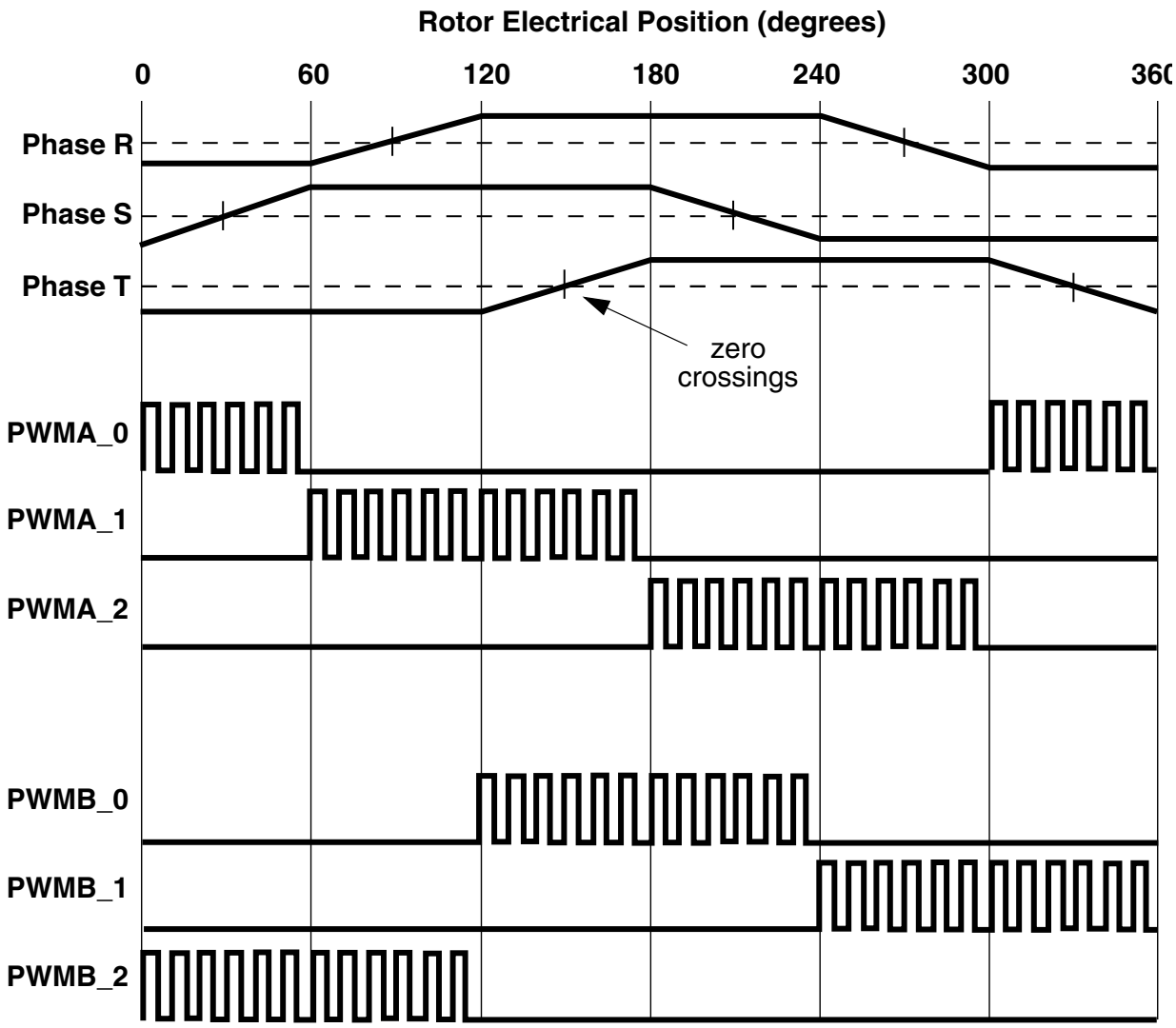


Figure 282. Sensorless BLDC commutation using the force out function

54.3.3 Operation

This section describes the implementation of various sections of the PWM in detail.

The following figure is a high-level block diagram of output PWM generation.

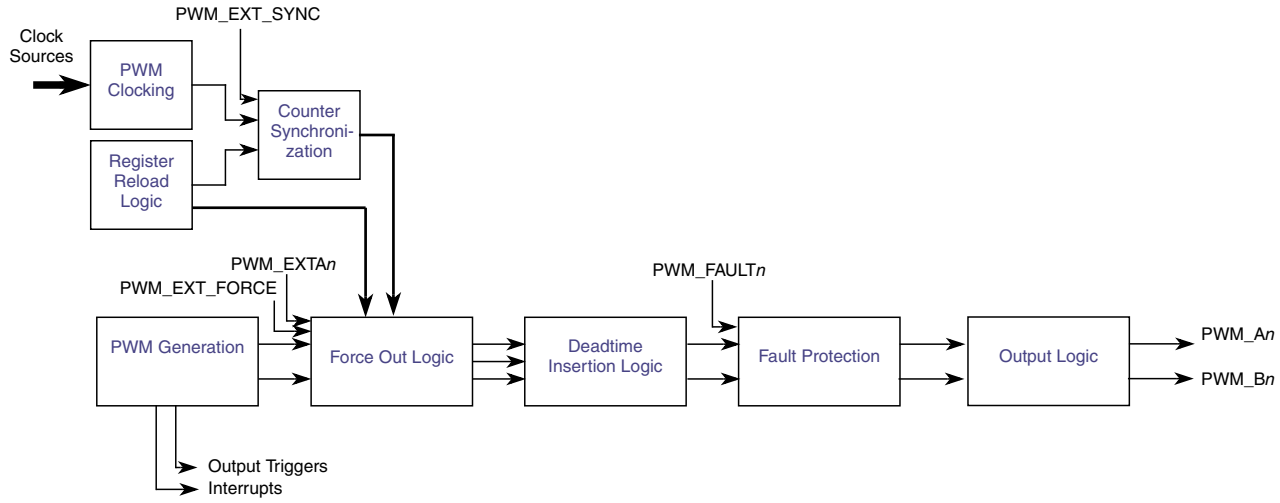


Figure 283. High-level output PWM generation block diagram

54.3.3.1 Register reload logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using CTRL[LDFQ] and CTRL[FULL], which is defined by VAL1 register. A half cycle reload option is also supported (CTRL[HALF]) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As shown in Figure 284 the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

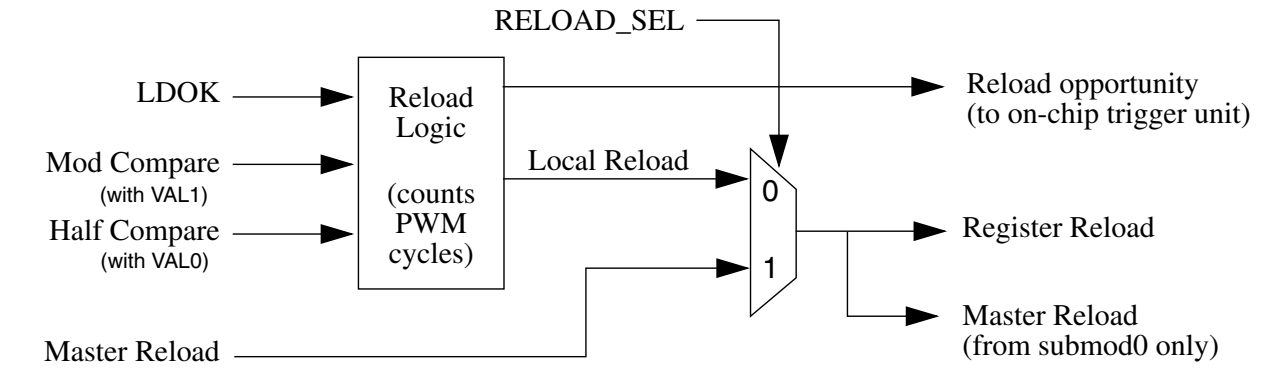


Figure 284. Register reload logic of each submodule

54.3.3.2 Counter synchronization

As shown in Figure 285, the 16-bit counter counts up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Synchronization signal which is one of four possible sources used to cause the 16-bit counter to be initialized with INIT. If Local Synchronization is selected as the counter initialization signal, VAL1 within the submodule effectively controls the timer period (and then the PWM frequency generated by that submodule), and everything operates or functions at a local level.

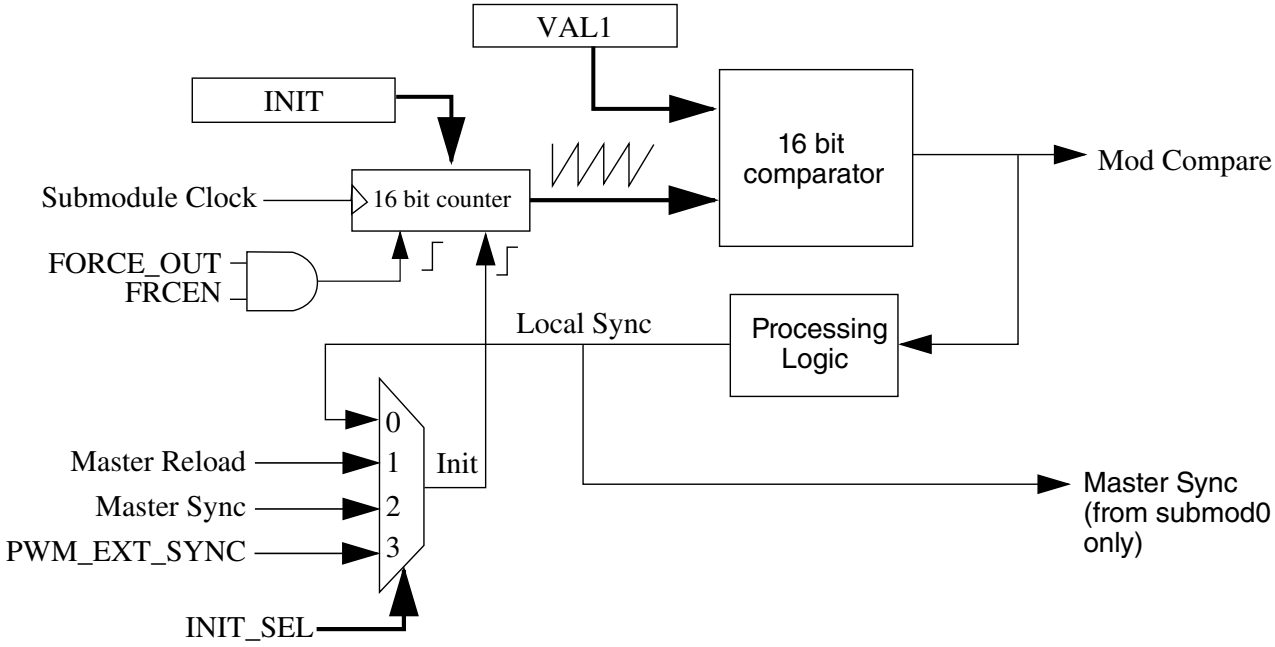


Figure 285. Counter synchronization for each submodule

The Master Synchronization signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0.

The PWM_EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is commensurate to the sampling frequency of the software control algorithm, the submodule counter period is equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that CTRL2[FRCEN] is set. As shown in Figure 285, this constitutes a second initialization input into the counter, which causes the counter to initialize regardless of which signal is selected as the counter initialization signal. A forced initialization causes a register reload if MCTRL[LDOK] is set.

The counter can be initialized by FORCE_OUT signal only when MCTRL[RUN] = 1 or CTLR2[CLK_SEL] = 2 which chooses auxiliary clock from SM0.

The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event for the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result is an oscillation caused by the beating between the PWM frequency and the commutation frequency.

54.3.3.3 PWM generation

Figure 286 illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register are used to control the turn-on edge, while a second comparator and VALx register control the turn-off edge.

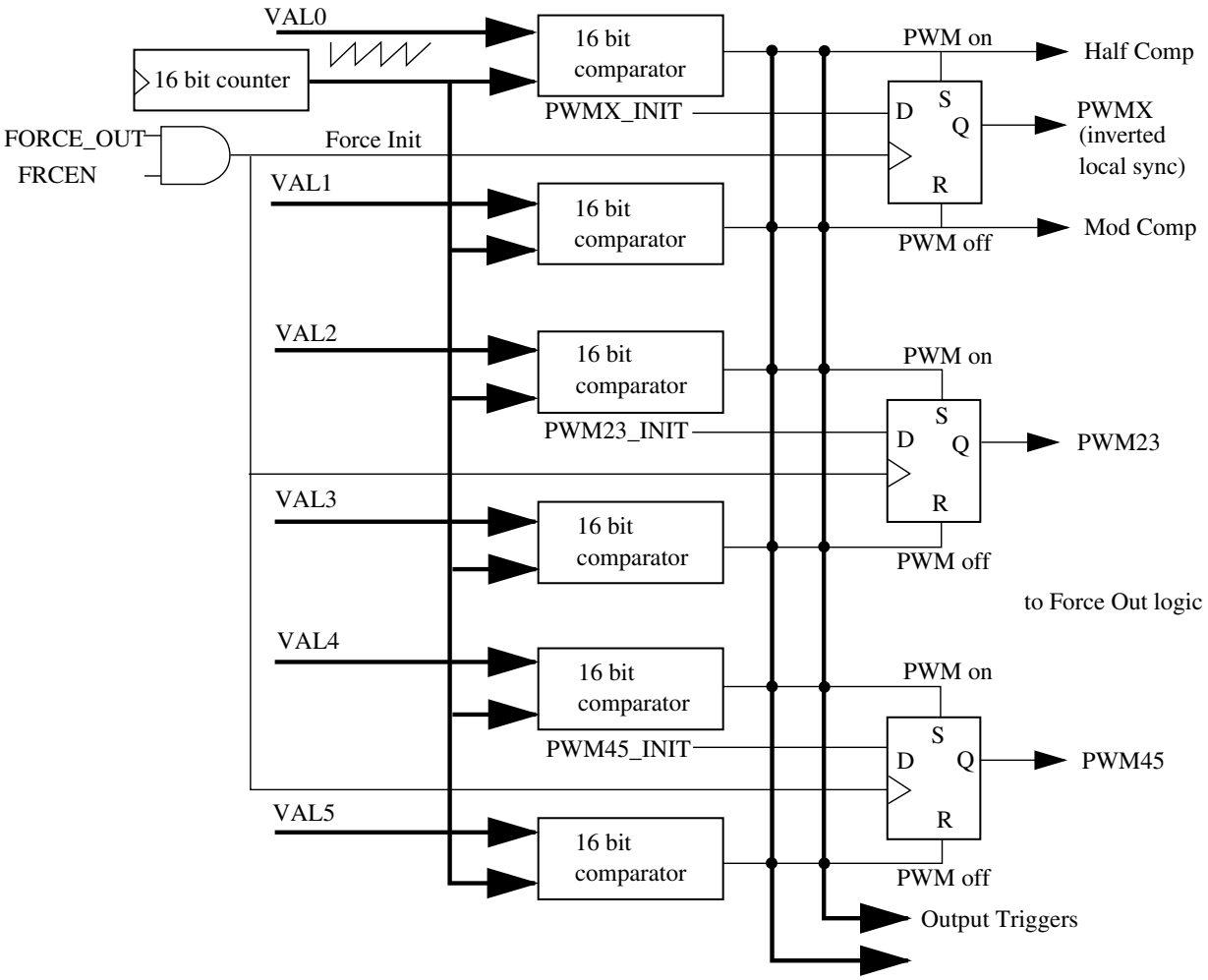


Figure 286. PWM Generation Hardware of each submodule

The generation of the Local Synchronization signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a falling edge of the Local Synchronization signal, comparator 1 generates a rising edge. Comparator 1 is also hardwired to the reload logic to generate the full cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half cycle reload pulse occurs exactly half way through the timer count period and the Local Synchronization will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Synchronization signal, effectively turning it into an auxiliary PWM signal (PWM_X) assuming that the PWM_X pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Synchronization signal, each submodule is capable of generating three PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The 16-bit comparators shown in Figure 286 are "equal to" comparators. In addition, if both the set and reset of the flip-flop are asserted, then the flop output goes to 0.

54.3.3.4 Output compare capabilities

By using the VALx registers in conjunction with the submodule timer and 16-bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

In PWM generation, an output compare is initiated by programming a VALx register for a timer compare, which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWM_A signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the VAL3 register must be programmed to a value out of the modulus range of the counter. Therefore, a comparison that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWM_A signal that sets it low, the VAL3 register is programmed with the appropriate count value and the VAL2 register is programmed with a value out of the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt, or output trigger can be generated when the compare event occurs.

54.3.3.5 Force out logic

For each submodule, the software can select between eight signal sources for the FORCE_OUT signal depending on the chip architecture:

1. Local CTRL2[FORCE]
2. Master Force signal from submodule0
3. Local Reload signal
4. Master Reload signal from submodule0
5. Local Synchronization signal
6. Master Synchronization signal from submodule0
7. EXT_SYNC signal from on or off chip
8. EXT_FORCE signal from on or off chip

The local signals are used to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master, EXT_SYNC, or EXT_FORCE signals must be selected.

Figure 287 illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the PWM_EXTA alternate external control signals. The selection can be determined ahead of time, and when a FORCE_OUT event occurs, these values are presented to the signal selection mux that immediately switches the requested signal to the output of the mux for further processing downstream.

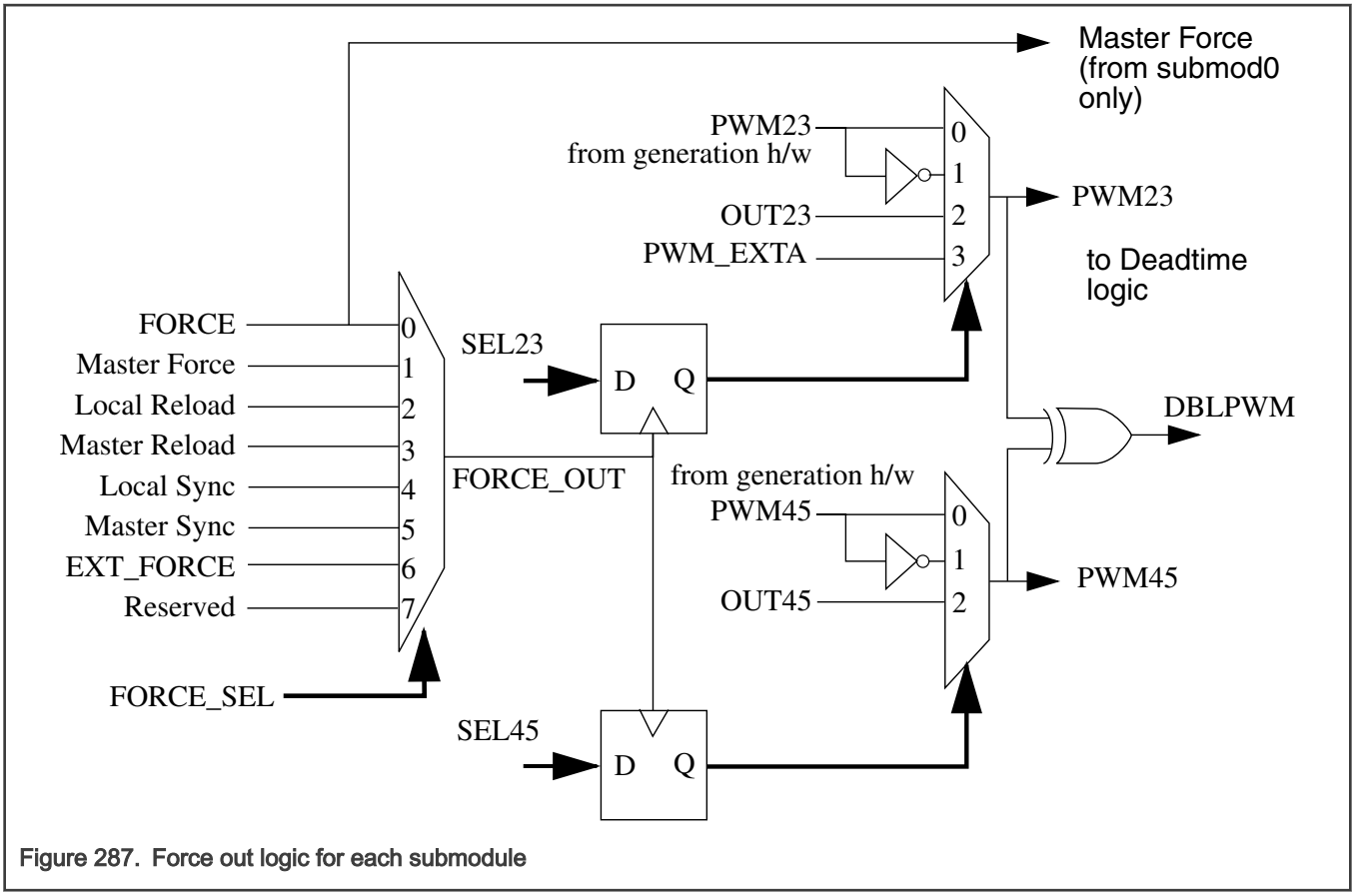


Figure 287. Force out logic for each submodule

The local CTRL2[FORCE] signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the CTRL2[FORCE] of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

54.3.3.6 Independent or complementary channel operation

Writing a logic one to CTRL2[INDEP] configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in Figure 3-16 in complementary channel operation. The signal that is connected to the output pin (PWM23 or PWM45) is determined by MCTRL[IPOL].

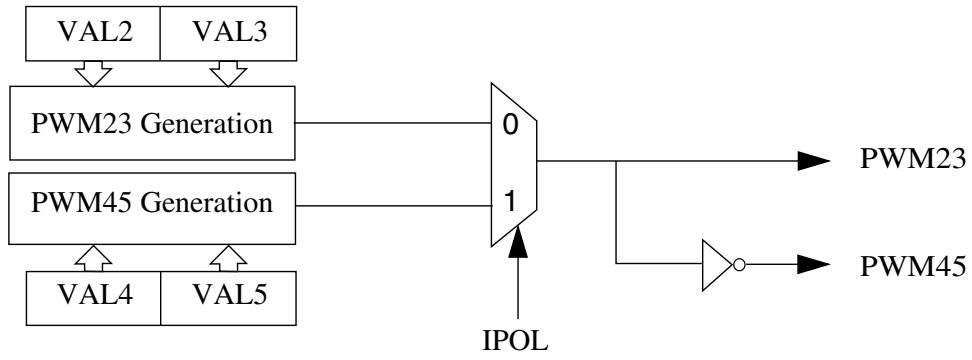


Figure 288. Complementary channel pair for each submodule

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, as shown in Figure 289.

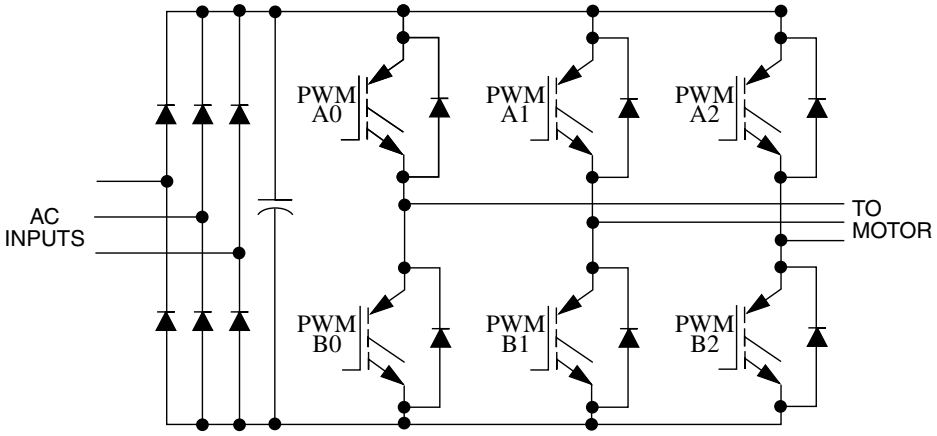


Figure 289. Typical 3 phase AC motor drive

The complementary operation allows the use of the deadtime insertion feature.

54.3.3.7 Deadtime insertion logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

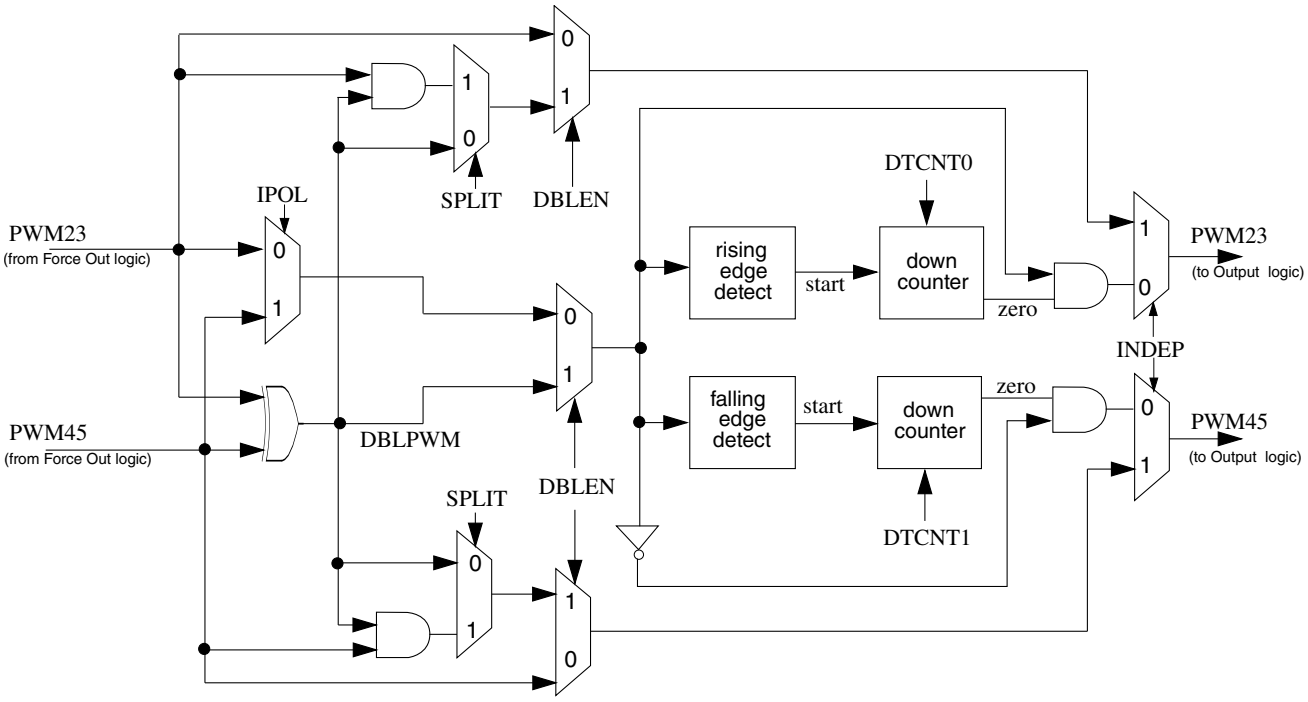


Figure 290. Deadtime Insertion Logic for each submodule

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

NOTE

To avoid short-circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between the top and bottom transistors. But the transistor's characteristics make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as shown in Figure 291.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

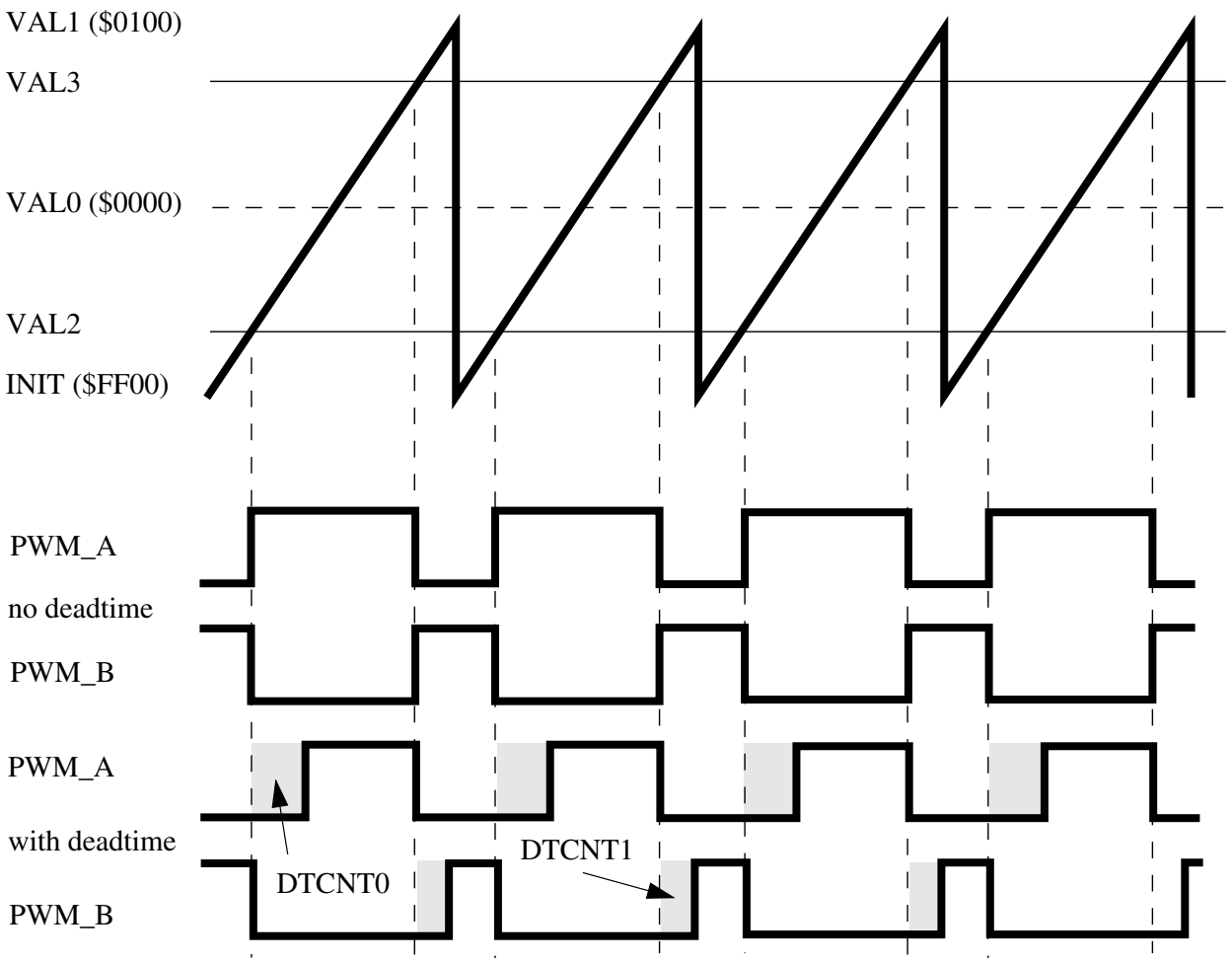


Figure 291. Deadtime insertion

54.3.3.7.1 Top/Bottom correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introducing distortion in the output voltage, as shown in Figure 292. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

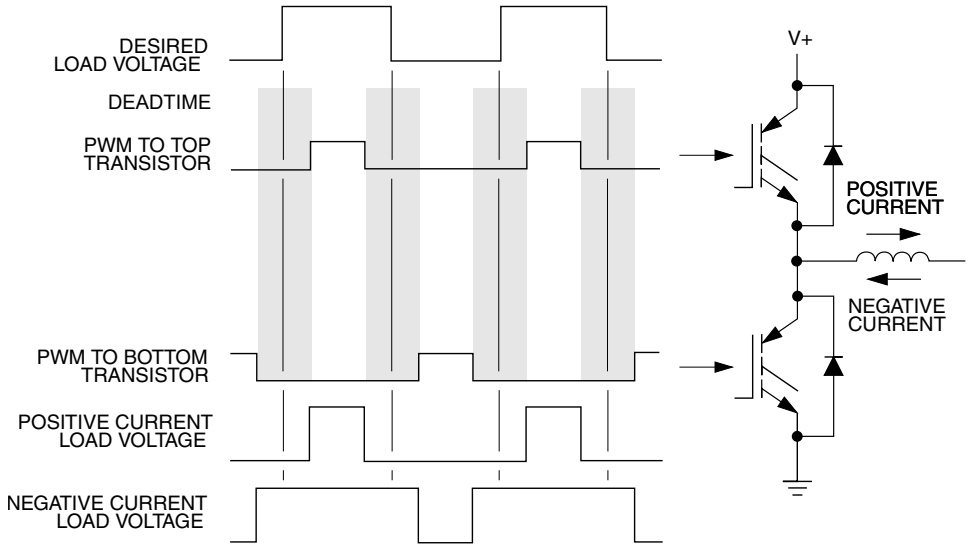


Figure 292. Deadtime distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with the current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths were shortened by deadtime insertion, the averaged sinusoidal output is less than the desired value. However, when deadtime is inserted, it distorts in the motor current waveform inverter outputs. This distortion is aggravated by different turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors is effective in controlling the output voltage at any given time. This depends on the direction of the motor inverter current for that pair, as shown in Figure 292. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the VALx registers. Either the VAL2/VAL3 or the VAL4/VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the VAL2/VAL3 or VAL4/VAL5 pair is active depends on either:

- The state of the current status pin, PWMX, for that driver
- The state of the odd/even correction bit, MCTRL[IPOL], for that driver

To correct deadtime distortion, the software can decrease or increase the value in the appropriate VALx register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

54.3.3.7.2 Manual correction

To detect the current status, the voltage on each PWMX pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in CTRL[DT]. CTRL[DT] is a timing marker indicating when to toggle between PWM value registers. The software can then set MCTRL[IPOL] to switch between VAL2/VAL3 and VAL4/VAL5 register pairs according to CTRL[DT] values.

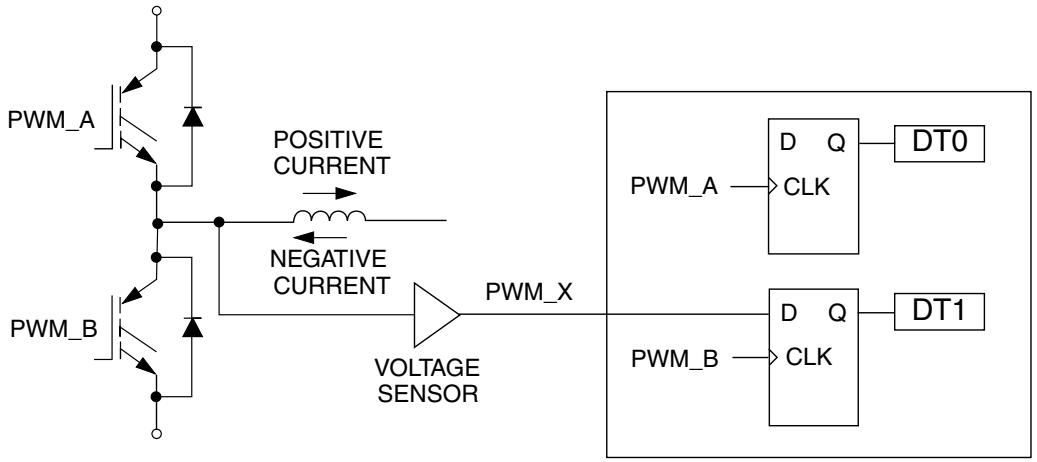


Figure 293. Current-status sense scheme for deadtime correction

Both D flip-flops latch low, CTRL[DT] = 00, during deadtime periods if the current is large and flowing out of the complementary circuit. See the preceding figure. Both D flip-flops latch the high, CTRL[DT] = 11, during deadtime periods if the current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. Sampled results are CTRL[DT] = b10. Thus, the best time to change one PWM value register to another is just before the current zero crossing.

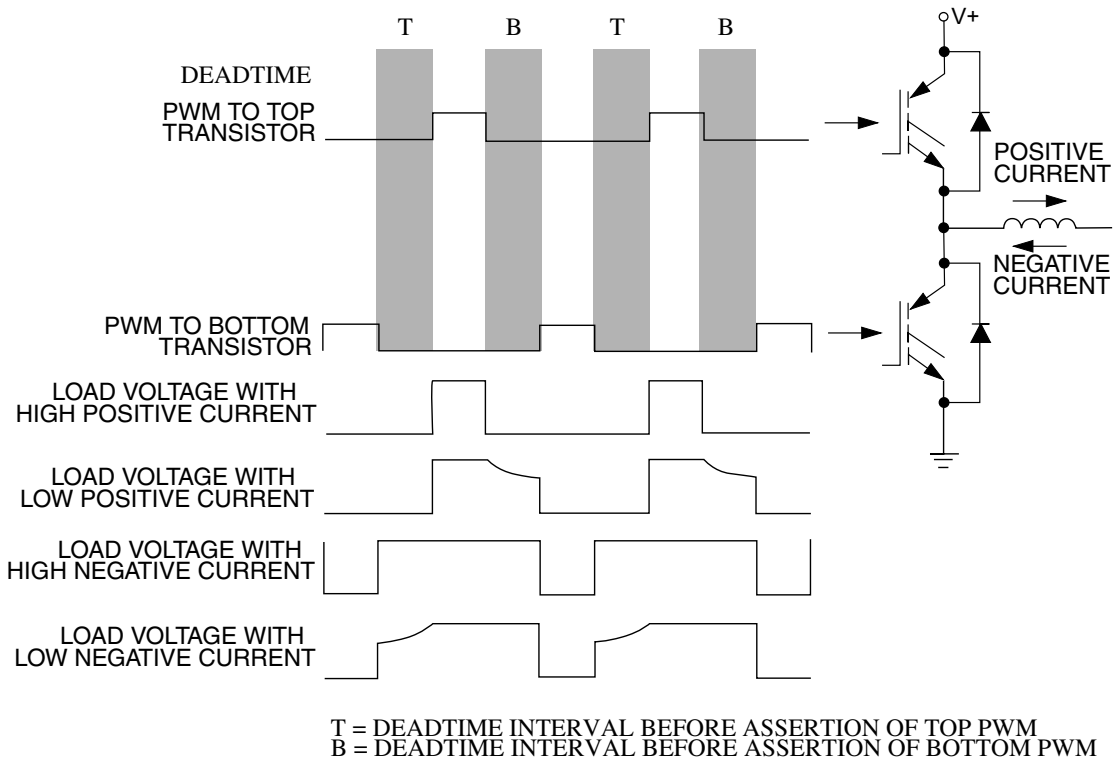


Figure 294. Output voltage waveforms

54.3.3.8 Fractional delay logic

For applications where more resolution than a single IPBus clock period is needed, the fractional delay logic can be used to achieve fine resolution on the rising and falling edges of the PWM_A and PWM_B outputs. Enable the use of the fractional delay logic by setting FRCTRL[FRACx_EN]. The FRACVALx registers act as a fractional clock cycle addition to the turn-on and turn-off count specified by the VAL2, VAL3, VAL4, or VAL5 registers. The FRACVAL1 register acts as a fractional increase in the PWM period as defined by VAL1. If FRACVAL1 is programmed to a non-zero value, then the largest value for the VAL1 register is 0xFFFE for unsigned usage or 0x7FFE for signed usage. This limit is required to avoid counter rollovers when accumulating the fractional additional period.

Both the fractional enables (1, 23, 45) and the fractional values (1-5) are double buffered and reloaded at the same time as the value registers.

Each PWM cycle, the value compare point is increased by 1 when there is overflow on the double buffered value of the fractional register plus the 5-bit accumulated fractional value. The accumulated fractional value starts at zero, so it is impossible to overflow the first PWM cycle.

At the end of each PWM cycle, if the corresponding fractional enable is set then the accumulated fractional value increments by the fractional value (double buffered value). The accumulated fractional value is 5-bits, so in the case of overflow only the remainder is kept. If the corresponding fractional enable is clear, then the accumulated fractional value is reset. This is the only way to reset the accumulated fractional value.

The accumulated fractional values are not accessible to software.

To fine-tune the PWM period using the FRACVAL1 register, if you want a period of 100.25 clock cycles, program VAL1 with 0x0064 and FRACVAL1 with 0x4000. The fractional value will accumulate so that every 4 PWM cycles will be 1 clock cycle longer (101 instead of 100). The rising and falling edges of the PWM outputs will also use the accumulated fraction to delay their edges and maintain a consistent 100.25 cycles spacing between corresponding edges from one cycle to the next.

The results of the fractional delay logic depend on whether or not the PWM submodule has an analog NanoEdge placer block available. With fractional delay enabled, PWM works in digital dithering mode: the value of FRACVALx is accumulated in each PWM period; when the accumulated value of FRACVALx overflows, VALx increments one in the next PWM period.

54.3.3.8.1 Fractional delay logic without NanoEdge placement block

For submodules that are not supported by the NanoEdge placer, the PWM can use dithering to simulate fine edge control. Enable this feature by setting the FRCTRL[FRAC1_EN], FRCTRL[FRAC23_EN], and FRCTRL[FRAC45_EN] bits. The PWM period or the PWM edges will dither from the nearest whole number values to achieve an average value that is equivalent to the programmed fractional value. The added cycles are based on the accumulation of the fractional component. For example, if you want the PWM period to be 50.25 clock cycles, then program VAL1 with 0x0032 and FRACVAL1 with 0x4000. The PWM period will be 50 cycles long most of the time, but will occasionally be 51 cycles long to achieve a long-term average of 50.25 cycles.

In submodules that are not supported by a NanoEdge placer, the clock frequency is not required to be any specific value to achieve proper operation.

54.3.3.9 Output logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing with the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic (as shown in Figure 295) are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program OCTRL[POLA] and OCTRL[POLB] before enabling the output pins. A fault condition can result in the PWM output being put in a high-impedance state, forced to a logic 1 state, or forced to a logic 0 state, depending on the values programmed into the OCTRL[PWMxFS] fields.

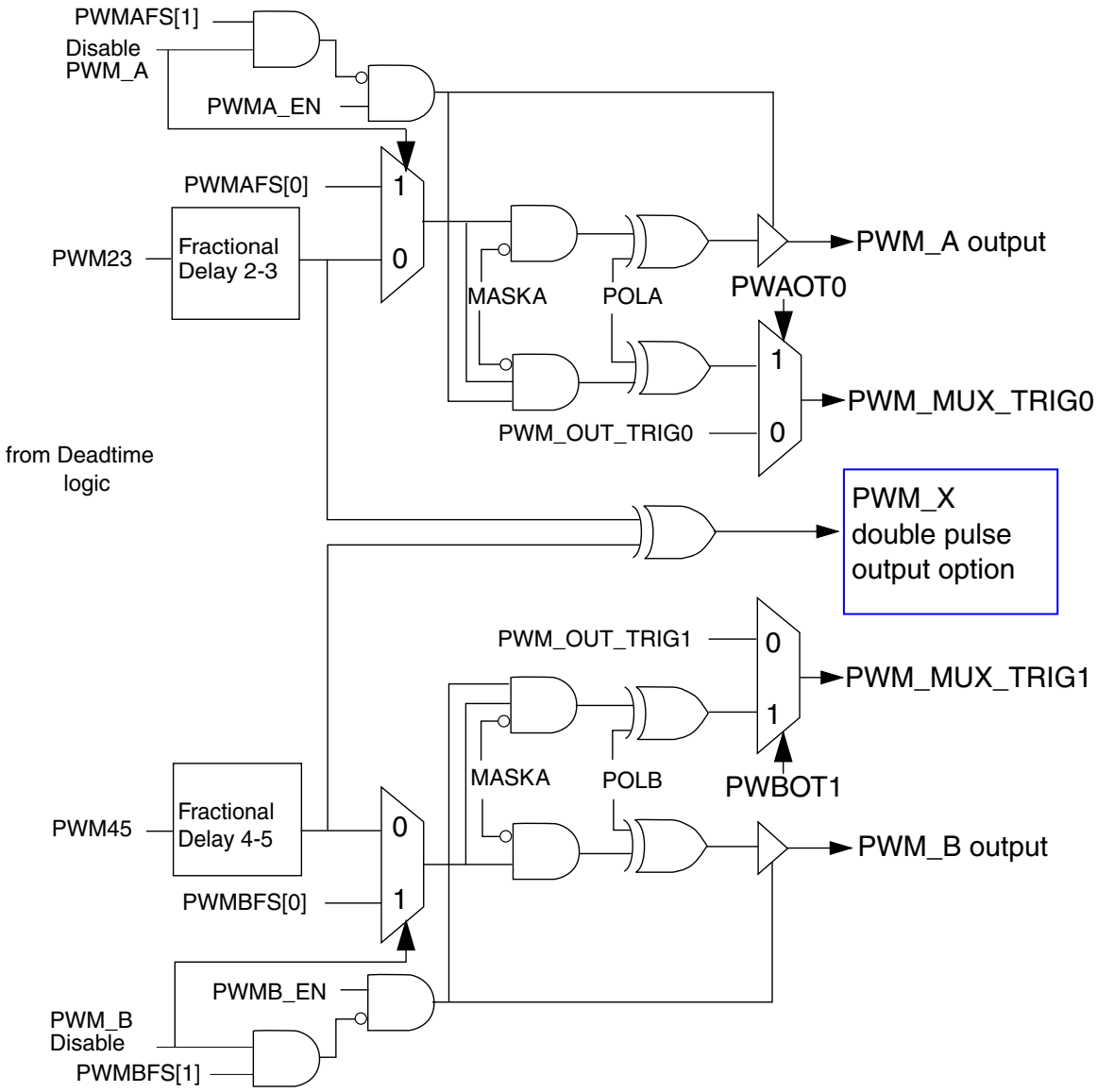


Figure 295. Output logic for each submodule

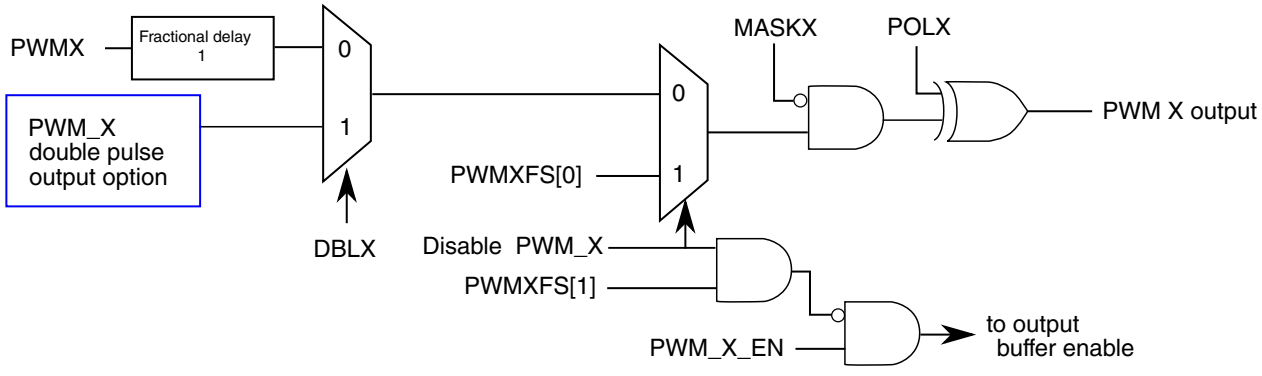


Figure 296. Output logic continued

54.3.3.10 E-Capture

The Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure shows block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8-bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8-bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. This feature is useful for dividing down high frequency signals for capture processing so that capture interrupts do not overwhelm the CPU. Also, this feature can be used to generate an interrupt after "n" events have been counted.

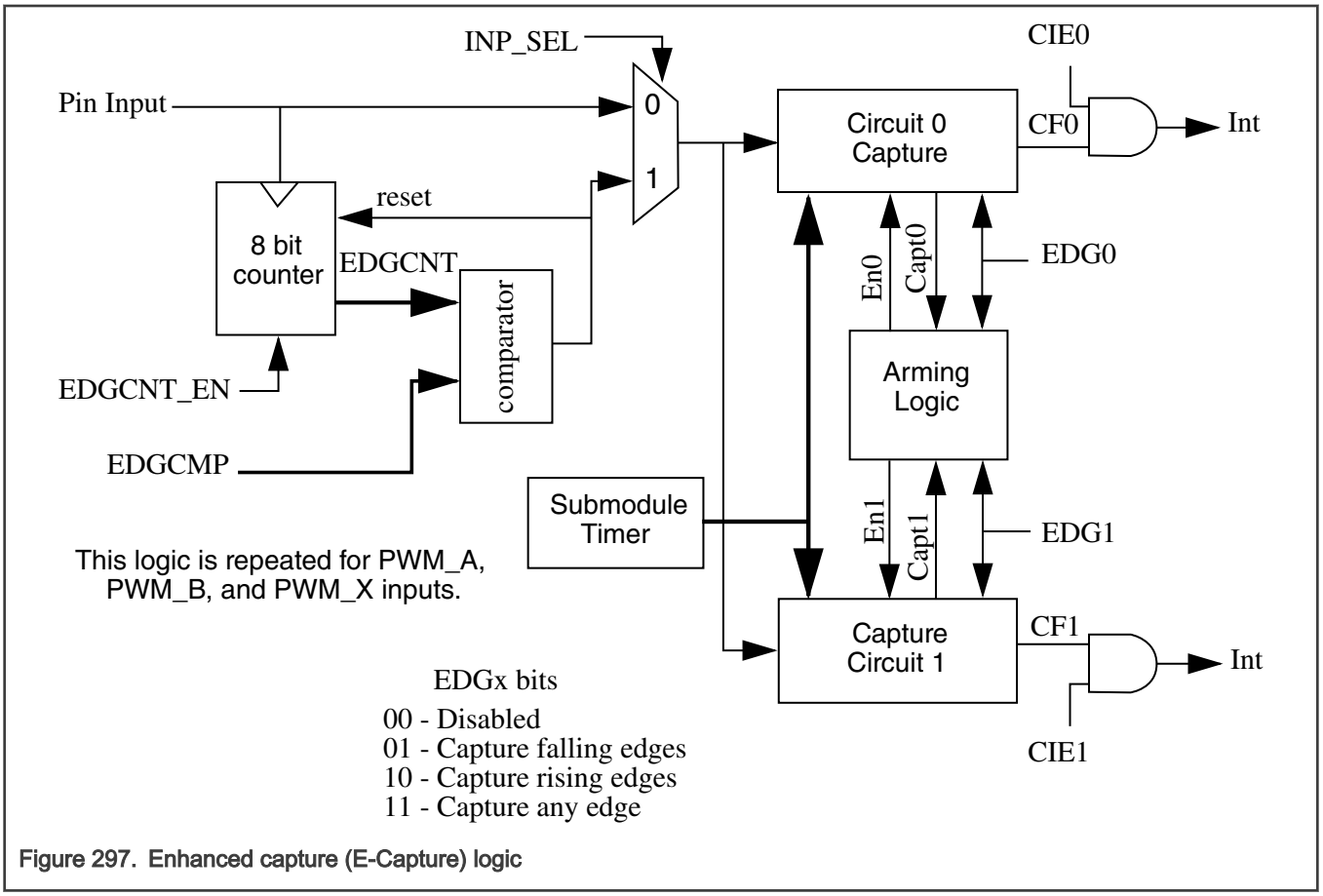


Figure 297. Enhanced capture (E-Capture) logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by CAPTCTRLx[EDGx1] and CAPTCTRLx[EDGx0], whose functionality is listed in the above figure. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot mode. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

54.3.3.11 Fault protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via FCTRL[FLVL]. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or high impedance depending on the values of OCTRL[PWMxFS].

The fault decoder disables PWM pins selected by the fault logic and the disable mapping (DISMAPn) registers. The following figure shows an example of the fault disable logic. Each bank of bits in DISMAPn control the mapping for a single PWM pin. See the following table.

The fault protection is enabled even when the PWM module is not enabled. Therefore, a fault is latched in and must be cleared to prevent an interrupt when the PWM is enabled.

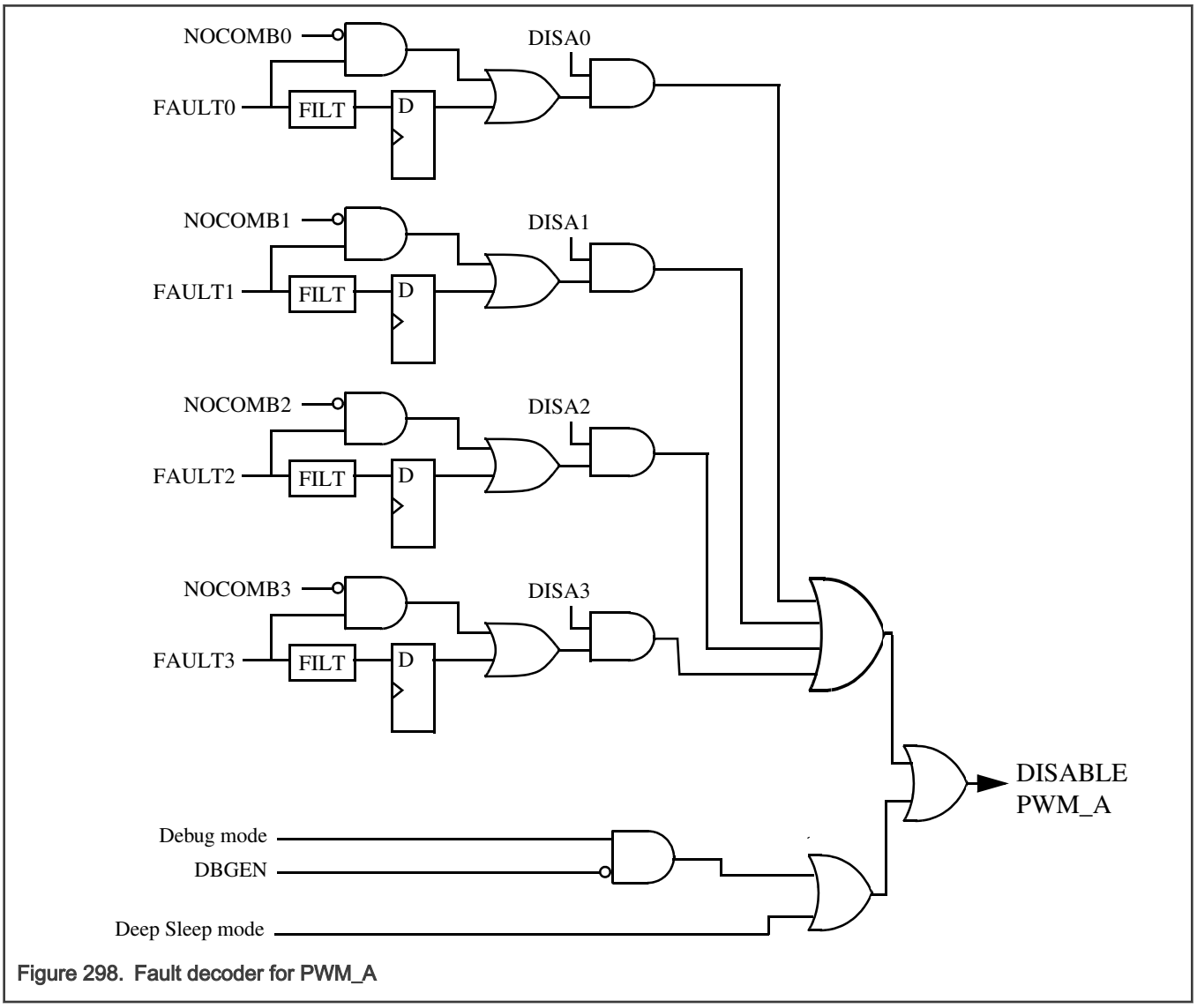


Figure 298. Fault decoder for PWM_A

Table 446. Fault Mapping

PWM Pin for 1 submodule	Controlling Register Bits
PWM_A	DISMAP0[DIS0A]

Table continues on the next page...

Table 446. Fault Mapping (continued)

PWM Pin for 1 submodule	Controlling Register Bits
PWM_B	DISMAP0[DIS0B]
PWM_X	DISMAP0[DIS0X]

54.3.3.11.1 Fault pin filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with FFILT[FILT_PER]. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using FFILT[FILT_CNT]. Setting FFILT[FILT_PER] to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FCTRL[FLVLx] is set), the corresponding FSTS[FFPINx] and FSTS[FFLAGx] bits are set. FSTS[FFPINx] remains set as long as the filtered FAULTx pin is zero. Clear FSTS[FFLAGx] by writing a logic 1 to FSTS[FFLAGx].

If the FIEx, FAULTx pin interrupt enable bit is set, FSTS[FFLAGx] generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears FSTS[FFLAGx] by writing a logic one to the bit
- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins, which in turn bypasses the filter when FCTRL20[NOCOMBx] = 0. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

54.3.3.11.2 Automatic fault clearing

Setting an automatic clearing mode bit, FCTRL[FAUTOx] configures faults from the FAULTx pin for automatic clearing.

When FCTRL[FAUTOx] is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FSTS[FFULLx] is set and the fault condition on FAULTx disappears, then the disabled PWM pins are enabled at the start of next full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle. Clearing FSTS[FFLAGx] does not affect disabled PWM pins when FCTRL[FAUTOx] is set.

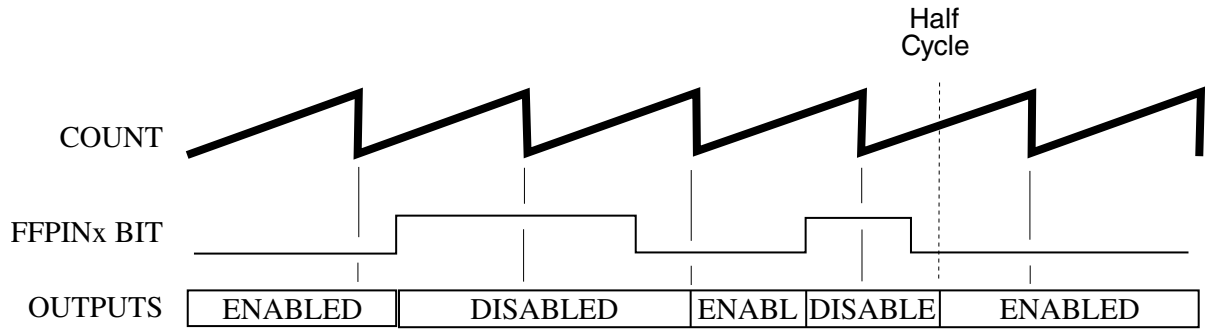


Figure 299. Automatic fault clearing

54.3.3.11.3 Manual fault clearing

Clearing the automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits FCTRL[FSAFEx] are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag

— The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin, as shown in Figure 300. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.

- If the fault safety mode bits FCTRL[FSAFEx] are set, then PWM pins disabled by the FAULTx pins are enabled when:
  — Software clears the corresponding FSTS[FFLAGx] flag
  — The filter detects a logic zero on the FAULTx pin at the start of the next PWM full or half cycle boundary, as shown in Figure 301. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.

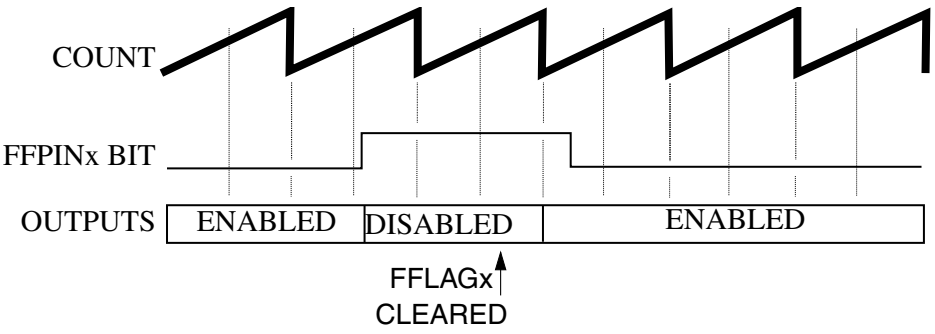


Figure 300. Manual fault clearing (FCTRL[FSAFEx] = 0, FSTS[FFULLx] = 1)

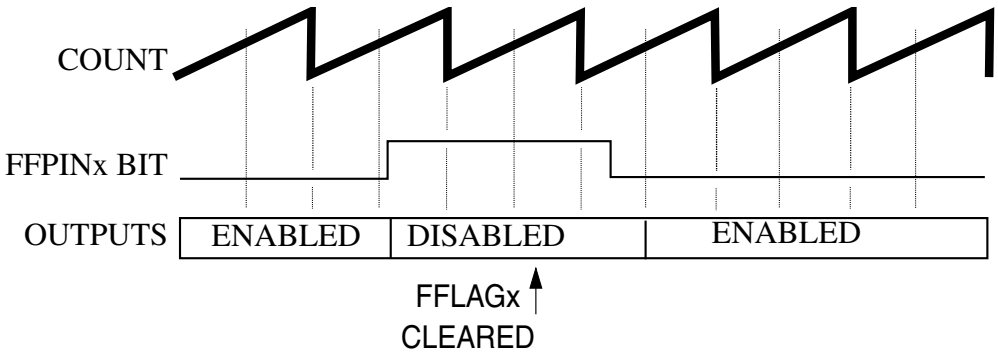


Figure 301. Manual fault clearing (FCTRL[FSAFEx] = 1, FSTS[FHALFx] = 1)

NOTE

Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or PWM_EXTA. Fault clearing still occurs at half or full PWM cycle boundaries while the PWM generator is engaged, MCTRL[RUN] equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, MCTRL[RUN] equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

54.3.3.11.4 Fault testing

FTST[FTEST] is used to simulate a fault condition on each of the fault inputs within that fault channel.

54.3.3.12 PWM generator loading

54.3.3.12.1 Load enable

MCTRL[LDOK] enables loading of the following PWM generator parameters.

- The prescaler divisor: from CTRL[PRSC]

- The PWM period and pulse width: from the INIT, FRACVALx, and VALx registers

MCTRL[LDOK] allows the software to finish calculating all of these PWM parameters so they can be synchronously updated. The CTRL[PRSC], INIT, and VALx registers are loaded by software into a set of outer buffers. When MCTRL[LDOK] is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator when CTRL[LDMOD] is cleared. These values can be transferred to the inner set of registers immediately upon setting MCTRL[LDOK] if CTRL[LDMOD] is set. Set MCTRL[LDOK] by reading it when it is a logic zero and then writing a logic one to it. After loading, MCTRL[LDOK] is automatically cleared.

54.3.3.12.2 Load frequency

CTRL[LDFQ] selects an integral loading frequency of one to 16 PWM reload opportunities. CTRL[LDFQ] takes effect at every PWM reload opportunity, regardless of the state of MCTRL[LDOK]. CTRL[HALF] and CTRL[FULL] control reload timing. If CTRL[FULL] is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If CTRL[HALF] is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both CTRL[HALF] and CTRL[FULL] are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

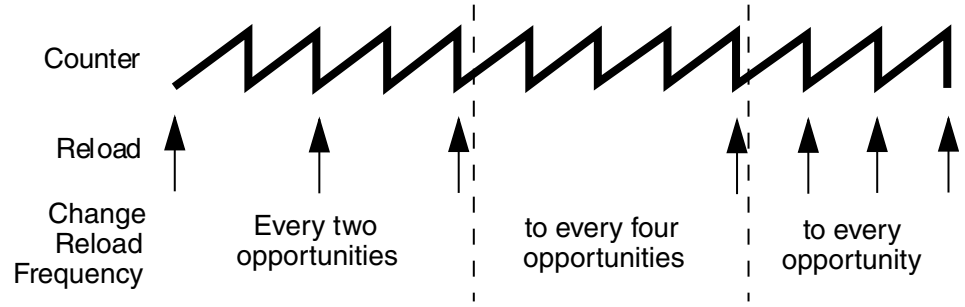


Figure 302. Full cycle reload frequency change

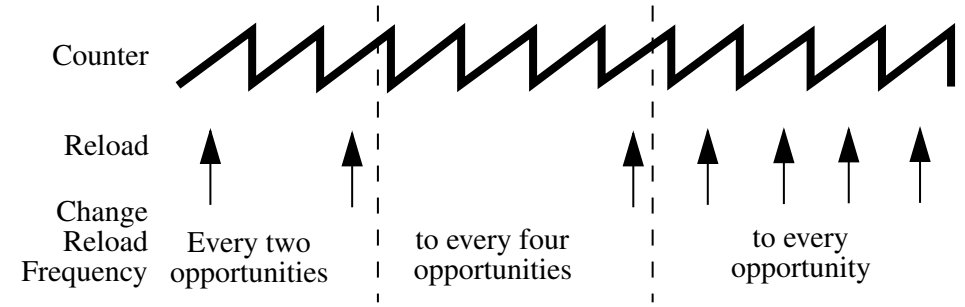


Figure 303. Half cycle reload frequency change

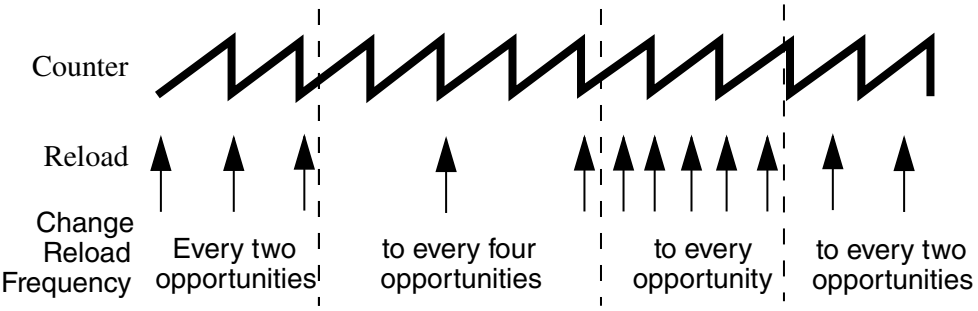


Figure 304. Full and half cycle reload frequency change

54.3.3.12.3 Reload flag

At every reload opportunity the PWM Reload Flag (STS[RF]) is set. Setting STS[RF] happens even if an actual reload is prevented by MCTRL[LDOK]. If the PWM reload interrupt enable bit INTEN[RIE] is set, the STS[RF] flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When INTEN[RIE] is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

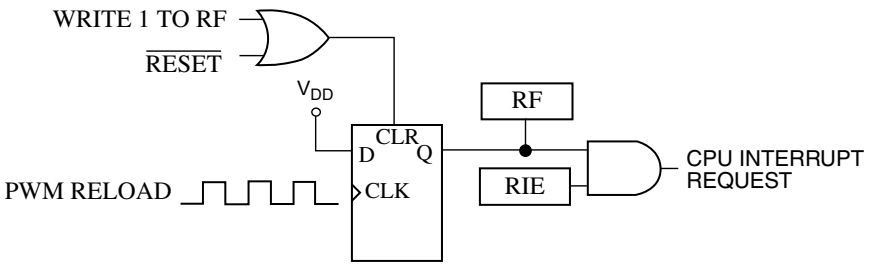


Figure 305. PWMF reload interrupt request

54.3.3.12.4 Reload errors

When one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers is updated (written by software), the STS[RUF] flag is set to indicate the data in the set of double buffered registers is not coherent. STS[RUF] is cleared by a successful reload which consists of the reload signal while MCTRL[LDOK] is set. If STS[RUF] is set and MCTRL[LDOK] is clear when the reload signal occurs, a reload error takes place and STS[REF] is set. If STS[RUF] is clear when a reload signal asserts, then the data is coherent and no error is flagged.

54.3.3.12.5 Initialization

Initialize all registers and then set MCTRL[LDOK] before setting MCTRL[RUN].

NOTE

If MCTRL[LDOK] is not set, setting MCTRL[RUN] also sets the STS[RF] flag. To prevent a CPU interrupt request, clear INTEN[RIE] before setting MCTRL[RUN].

The PWM generator uses the last values loaded if MCTRL[RUN] is cleared and then set while MCTRL[LDOK] equals zero.

When MCTRL[RUN] is cleared:

- The STS[RF] flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software/external output control remains active
- Deadtime insertion continues during software/external output control

54.3.4 Power modes

Be careful when using this module in Deep Sleep, Sleep, and Debug operating modes.

CAUTION

Some applications require regular software updates for proper operation. Failure to provide regular software updates could result in destroying the hardware setup.

To accommodate this situation, PWM outputs are placed in their inactive states in Deep Sleep mode, and they can optionally be placed in inactive states in Sleep and Debug modes. PWM outputs are reactivated (assuming they were active beforehand) when these modes are exited.

Table 447. Modes when PWM operation is restricted

Mode	Description
Deep Sleep	PWM outputs are inactive.
Debug	PWM outputs are driven or inactive as a function of CTRL2[DBGEN].

54.3.5 Clocking

Figure 306 shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the IPBus clock, EXT_CLK, and AUX_CLK. The EXT_CLK goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8-bit prescaler and MCTRL[RUN] from submodule0 can control all of the submodules.

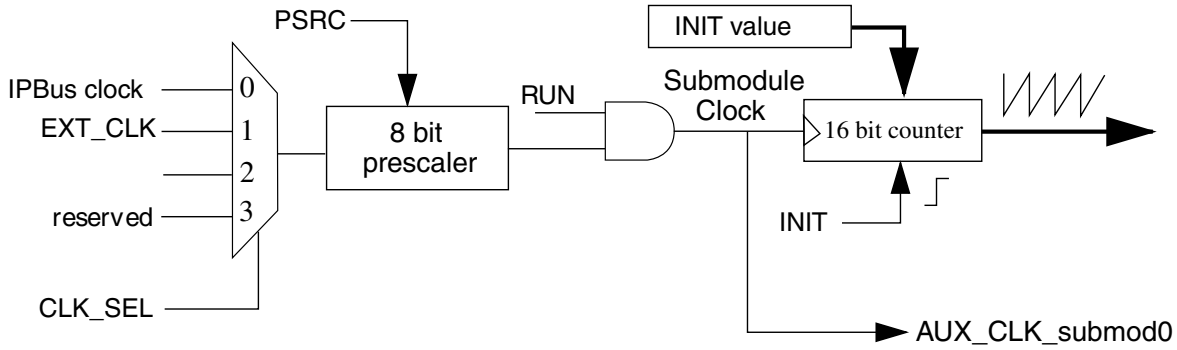


Figure 306. Clocking Block Diagram for PWM Submodule 0

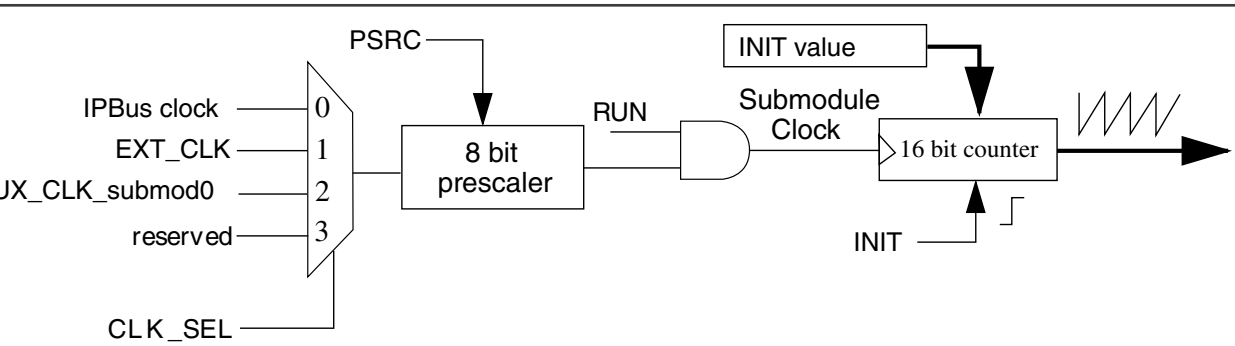


Figure 307. Clocking Block Diagram for all other PWM Submodules

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by 1-128. The prescaler bits, CTRL[PRSC], select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until MCTRL[LDOK] is set and a new PWM reload cycle begins or CTRL[LDMOD] is set.

54.3.6 Resets

All PWM registers are reset to their default values upon any system reset.

The reset forces all registers to their reset states and tristates the PWM outputs.

54.3.7 Interrupts

Each of the submodules within the eFlexPWM module can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 448. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CMP0	SM0STS[CMPF]	SM0INTEN[CMPIE]	Submodule 0 compare interrupt	Compare event has occurred
PWM_CAP0	SM0STS[CFA1], SM0STS[CFA0], SM0STS[CFB1], SM0STS[CFB0], SM0STS[CFX1], SM0STS[CFX0]	SM0INTEN[CFA1IE], SM0INTEN[CFA0IE], SM0INTEN[CFB1IE], SM0INTEN[CFB0IE], SM0INTEN[CFX1IE], SM0INTEN[CFX0IE]	Submodule 0 input capture interrupt	Input capture event has occurred
PWM_RELOAD0	SM0STS[RF]	SM0INTEN[RIE]	Submodule 0 reload interrupt	Reload event has occurred
PWM_CMP1	SM1STS[CMPF]	SM1INTEN[CMPIE]	Submodule 1 compare interrupt	Compare event has occurred
PWM_CAP1	SM1STS[CFA1], SM1STS[CFA0], SM1STS[CFB1], SM1STS[CFB0], SM1STS[CFX1], SM1STS[CFX0]	SM1INTEN[CFA1IE], SM1INTEN[CFA0IE], SM1INTEN[CFB1IE], SM1INTEN[CFB0IE], SM1INTEN[CFX1IE], SM1INTEN[CFX0IE]	Submodule 1 input capture interrupt	Input capture event has occurred
PWM_RELOAD1	SM1STS[RF]	SM1INTEN[RIE]	Submodule 1 reload interrupt	Reload event has occurred
PWM_CMP2	SM2STS[CMPF]	SM2INTEN[CMPIE]	Submodule 2 compare interrupt	Compare event has occurred
PWM_CAP2	SM2STS[CFA1], SM2STS[CFA0], SM2STS[CFB1], SM2STS[CFB0], SM2STS[CFX1], SM2STS[CFX0]	SM2INTEN[CFA1IE], SM2INTEN[CFA0IE], SM2INTEN[CFB1IE], SM2INTEN[CFB0IE], SM2INTEN[CFX1IE], SM2INTEN[CFX0IE	Submodule 2 input capture interrupt	Input capture event has occurred
PWM_RELOAD2	SM2STS[RF]	SM2INTEN[RIE]	Submodule 2 reload interrupt	Reload event has occurred

Table continues on the next page...

Table 448. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CMP3	SM3STS[CMPF]	SM3INTEN[CMPIE]	Submodule 3 compare interrupt	Compare event has occurred
PWM_CAP3	SM3STS[CFA1], SM3STS[CFA0], SM3STS[CFB1], SM3STS[CFB0], SM3STS[CFX1], SM3STS[CFX0]	SM3INTEN[CFA1IE], SM3INTEN[CFA0IE], SM3INTEN[CFB1IE], SM3INTEN[CFB0IE], SM3INTEN[CFX1IE], SM3INTEN[CFX0IE	Submodule 3 input capture interrupt	Input capture event has occurred
PWM_RELOAD3	SM3STS[RF]	SM3INTEN[RIE]	Submodule 3 reload interrupt	Reload event has occurred
PWM_RERR	SM0STS[REF]	SM0INTEN[REIE]	Submodule 0 reload error interrupt	Reload error has occurred
	SM1STS[REF]	SM1INTEN[REIE]	Submodule 1 reload error interrupt	
	SM2STS[REF]	SM2INTEN[REIE]	Submodule 2 reload error interrupt	
	SM3STS[REF]	SM3INTEN[REIE]	Submodule 3 reload error interrupt	
PWM_FAULT	FSTS[FFLAG]	FCTRL[FIE]	Fault input interrupt	Fault condition has been detected

54.3.8 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered registers.

Table 449. DMA summary

DMA request	DMA enable	Name	Description
Submodule 0 read request	SM0DMAEN[CX0DE]	SM0 Capture FIFO X0 read request	SM0CVAL0 contains a value to be read
Submodule 0 read request	SM0DMAEN[CX1DE]	SM0 Capture FIFO X1 read request	SM0CVAL1 contains a value to be read
Submodule 0 read request	SM0DMAEN[CA0DE]	SM0 Capture FIFO A0 read request	SM0CVAL2 contains a value to be read
Submodule 0 read request	SM0DMAEN[CA1DE]	SM0 Capture FIFO A1 read request	SM0CVAL3 contains a value to be read

Table continues on the next page...

Table 449. DMA summary (continued)

DMA request	DMA enable	Name	Description
Submodule 0 read request	SM0DMAEN[CB0DE]	SM0 Capture FIFO B0 read request	SM0CVAL4 contains a value to be read
Submodule 0 read request	SM0DMAEN[CB1DE]	SM0 Capture FIFO B1 read request	SM0CVAL5 contains a value to be read
Submodule 0 read request	SM0DMAEN[CAPTDE]	SM0 Capture FIFO read request source select	Selects source of submodule0 read DMA request
Submodule 0 write request	SM0DMAEN[VALDE]	SM0VALx write request	SM0VALx and SM0FRACVALx registers need to be updated
Submodule 1 read request	SM1DMAEN[CX0DE]	SM1 Capture FIFO X0 read request	SM1CVAL0 contains a value to be read
Submodule 1 read request	SM1DMAEN[CX1DE]	SM1 Capture FIFO X1 read request	SM1CVAL1 contains a value to be read
Submodule 1 read request	SM1DMAEN[CA0DE]	SM1Capture FIFO A0 read request	SM1CVAL2 contains a value to be read
Submodule 1 read request	SM1DMAEN[CA1DE]	SM1 Capture FIFO A1 read request	SM1CVAL3 contains a value to be read
Submodule 1 read request	SM1DMAEN[CB0DE]	SM1 Capture FIFO B0 read request	SM1CVAL4 contains a value to be read
Submodule 1 read request	SM1DMAEN[CB1DE]	SM1 Capture FIFO B1 read request	SM1CVAL5 contains a value to be read
Submodule 1 read request	SM1DMAEN[CAPTDE]	SM1 Capture FIFO read request source select	Selects source of submodule1 read DMA request
Submodule 1 write request	SM1DMAEN[VALDE]	SM1VALx write request	SM1VALx and SM1FRACVALx registers need to be updated
Submodule 2 read request	SM2DMAEN[CX0DE]	SM2 Capture FIFO X0 read request	SM2CVAL0 contains a value to be read
Submodule 2 read request	SM2DMAEN[CX1DE]	SM2 Capture FIFO X1 read request	SM2CVAL1 contains a value to be read
Submodule 2 read request	SM2DMAEN[CA0DE]	SM2 Capture FIFO A0 read request	SM2CVAL2 contains a value to be read
Submodule 2 read request	SM2DMAEN[CA1DE]	SM2 Capture FIFO A1 read request	SM2CVAL3 contains a value to be read
Submodule 2 read request	SM2DMAEN[CB0DE]	SM2 Capture FIFO B0 read request	SM2CVAL4 contains a value to be read

Table continues on the next page...

Table 449. DMA summary (continued)

DMA request	DMA enable	Name	Description
Submodule 2 read request	SM2DMAEN[CB1DE]	SM2 Capture FIFO B1 read request	SM2CVAL5 contains a value to be read
Submodule 2 read request	SM2DMAEN[CAPTDE]	SM2 Capture FIFO read request source select	Selects source of submodule2 read DMA request
Submodule 2 write request	SM2DMAEN[VALDE]	SM2VALx write request	SM2VALx and SM2FRACVALx registers need to be updated
Submodule 3 read request	SM3DMAEN[CX0DE]	SM3 Capture FIFO X0 read request	SM3CVAL0 contains a value to be read
Submodule 3 read request	SM3DMAEN[CX1DE]	SM3 Capture FIFO X1 read request	SM3CVAL1 contains a value to be read
Submodule 3 read request	SM3DMAEN[CA0DE]	SM3 Capture FIFO A0 read request	SM3CVAL2 contains a value to be read
Submodule 3 read request	SM3DMAEN[CA1DE]	SM3 Capture FIFO A1 read request	SM3CVAL3 contains a value to be read
Submodule 3 read request	SM3DMAEN[CB0DE]	SM3 Capture FIFO B0 read request	SM3CVAL4 contains a value to be read
Submodule 3 read request	SM3DMAEN[CB1DE]	SM3 Capture FIFO B1 read request	SM3CVAL5 contains a value to be read
Submodule 3 read request	SM3DMAEN[CAPTDE]	SM3 Capture FIFO read request source select	Selects source of submodule3 read DMA request
Submodule 3 write request	SM3DMAEN[VALDE]	SM3VALx write request	SM3VALx and SM3FRACVALx registers need to be updated

54.4 External signals

The PWM has pins named PWM_An, PWM_Bn, PWM_Xn, FAULTn, EXT_SYNC, EXT_FORCE, and PWMn_EXTA. The PWM also has an on-chip input called EXT_CLK and output signals called PWMn_OUT_TRIGx and PWMn_MUX_TRIGx.

54.4.1 PWM_An and PWM_Bn - External PWM output pair

These pins are the output pins of the PWM channels. These pins can be independent PWM signals or a complementary pair. When not needed as an output, they can be used as inputs to the input capture circuitry.

54.4.2 PWM_Xn - Auxiliary PWM output signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or used to detect the polarity of the current flowing through the complementary circuit at deadtime correction.

54.4.3 FAULTn - Fault Inputs

These are input pins for disabling selected PWM outputs.

54.4.4 EXT_SYNC - External synchronization signal

These input signals allow a source external to the PWM to initialize the PWM counter. Therefore, the PWM can be synchronized to external circuitry.

54.4.5 EXT_FORCE - External output force signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. Therefore, the PWM can be synchronized to external circuitry.

54.4.6 PWMn_EXTA - Alternate PWM control signals

These pins allow an alternate source to control the PWM_An and PWM_Bn outputs. Typically, the PWMn_EXTA input (depending on the state of MCTRL[IPOL]) is used for the generation of a complementary pair. Typical control signals include ADC conversion high/low limits, TMR outputs, GPIO inputs, and comparator outputs.

For pin input details, see chip-specific eFlexPWM information.

54.4.7 PWMn_OUT_TRIG0 and PWMn_OUT_TRIG1 - Output triggers

These outputs allow the PWM submodules to control timing of ADC conversions. See the description of the SMnTCTRL[OUT_TRIG_EN] for information about how to enable these outputs and how the compare registers match up to the output triggers.

54.4.8 PWM[n]_MUX_TRIG0 and PWM[n]_MUX_TRIG1 - Output triggers

These outputs can be either PWMn_OUT_TRIG0/PWMn_OUT_TRIG1 signals or PWM_A/PWM_B signals from output logic. See the description of the PWAOT0 and PWBOT1 bits in the Output Trigger Control Register for information about how to enable these outputs.

54.4.9 EXT_CLK - External clock signal

This signal allows a source external to the PWM (typically a timer or an off-chip source) to control the PWM clocking. Therefore, the PWM can be synchronized to the timer, or multiple chips can be synchronized to each other.

54.5 PWM register descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level, and the address offset is defined at the module level. The PWM module has a set of registers for each PWM submodule, for the configuration logic, and for each fault channel.

NOTE

While the registers are 16-bit wide, they can be accessed in pairs as 32-bit registers, if the pairs are word-aligned.

Submodule registers are repeated for each PWM submodule. To designate which submodule that they are in, register names are prefixed with SMx(x=0,1...). The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is offset 0x60 from the base address for the PWM module as a whole. This 0x60 offset is based on the number of registers in a submodule. The base address of submodule 2 is equal to the base address of submodule 1 plus this same 0x60 offset. The pattern repeats for the base address of submodule 3.

The base address of the configuration registers is equal to the base address of the PWM module as a whole plus an offset of 0x180.

54.5.1 PWM memory map

PWM0 base address: 400C_E000h

PWM1 base address: 400D_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Counter Register (SM0CNT)	16	R	0000h
2h	Initial Count Register (SM0INIT)	16	RW	0000h
4h	Control 2 Register (SM0CTRL2)	16	RW	0000h
6h	Control Register (SM0CTRL)	16	RW	0400h
Ah	Value Register 0 (SM0VAL0)	16	RW	0000h
Ch	Fractional Value Register 1 (SM0FRACVAL1)	16	RW	0000h
Eh	Value Register 1 (SM0VAL1)	16	RW	0000h
10h	Fractional Value Register 2 (SM0FRACVAL2)	16	RW	0000h
12h	Value Register 2 (SM0VAL2)	16	RW	0000h
14h	Fractional Value Register 3 (SM0FRACVAL3)	16	RW	0000h
16h	Value Register 3 (SM0VAL3)	16	RW	0000h
18h	Fractional Value Register 4 (SM0FRACVAL4)	16	RW	0000h
1Ah	Value Register 4 (SM0VAL4)	16	RW	0000h
1Ch	Fractional Value Register 5 (SM0FRACVAL5)	16	RW	0000h
1Eh	Value Register 5 (SM0VAL5)	16	RW	0000h
20h	Fractional Control Register (SM0FRCTRL)	16	RW	0000h
22h	Output Control Register (SM0OCTRL)	16	RW	0000h
24h	Status Register (SM0STS)	16	RW	0000h
26h	Interrupt Enable Register (SM0INTEN)	16	RW	0000h
28h	DMA Enable Register (SM0DMAEN)	16	RW	0000h
2Ah	Output Trigger Control Register (SM0TCTRL)	16	RW	0000h
2Ch	Fault Disable Mapping Register 0 (SM0DISMAP0)	16	RW	FFFFh
30h	Deadtime Count Register 0 (SM0DTCNT0)	16	RW	07FFh
32h	Deadtime Count Register 1 (SM0DTCNT1)	16	RW	07FFh
34h	Capture Control A Register (SM0CAPTCTRLA)	16	RW	0000h
36h	Capture Compare A Register (SM0CAPTCOMPA)	16	RW	0000h
38h	Capture Control B Register (SM0CAPTCTRLB)	16	RW	0000h
3Ah	Capture Compare B Register (SM0CAPTCOMPB)	16	RW	0000h
3Ch	Capture Control X Register (SM0CAPTCTRLX)	16	RW	0000h
3Eh	Capture Compare X Register (SM0CAPTCOMPX)	16	RW	0000h
40h	Capture Value 0 Register (SM0CVAL0)	16	R	0000h
42h	Capture Value 0 Cycle Register (SM0CVAL0CYC)	16	R	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
44h	Capture Value 1 Register (SM0CVAL1)	16	R	0000h
46h	Capture Value 1 Cycle Register (SM0CVAL1CYC)	16	R	0000h
48h	Capture Value 2 Register (SM0CVAL2)	16	R	0000h
4Ah	Capture Value 2 Cycle Register (SM0CVAL2CYC)	16	R	0000h
4Ch	Capture Value 3 Register (SM0CVAL3)	16	R	0000h
4Eh	Capture Value 3 Cycle Register (SM0CVAL3CYC)	16	R	0000h
50h	Capture Value 4 Register (SM0CVAL4)	16	R	0000h
52h	Capture Value 4 Cycle Register (SM0CVAL4CYC)	16	R	0000h
54h	Capture Value 5 Register (SM0CVAL5)	16	R	0000h
56h	Capture Value 5 Cycle Register (SM0CVAL5CYC)	16	R	0000h
5Ah	Capture PWM_A Input Filter Register (SM0CAPTFILTA)	16	RW	0000h
5Ch	Capture PWM_B Input Filter Register (SM0CAPTFILTB)	16	RW	0000h
5Eh	Capture PWM_X Input Filter Register (SM0CAPTFILTX)	16	RW	0000h
60h	Counter Register (SM1CNT)	16	R	0000h
62h	Initial Count Register (SM1INIT)	16	RW	0000h
64h	Control 2 Register (SM1CTRL2)	16	RW	0000h
66h	Control Register (SM1CTRL)	16	RW	0400h
6Ah	Value Register 0 (SM1VAL0)	16	RW	0000h
6Ch	Fractional Value Register 1 (SM1FRACVAL1)	16	RW	0000h
6Eh	Value Register 1 (SM1VAL1)	16	RW	0000h
70h	Fractional Value Register 2 (SM1FRACVAL2)	16	RW	0000h
72h	Value Register 2 (SM1VAL2)	16	RW	0000h
74h	Fractional Value Register 3 (SM1FRACVAL3)	16	RW	0000h
76h	Value Register 3 (SM1VAL3)	16	RW	0000h
78h	Fractional Value Register 4 (SM1FRACVAL4)	16	RW	0000h
7Ah	Value Register 4 (SM1VAL4)	16	RW	0000h
7Ch	Fractional Value Register 5 (SM1FRACVAL5)	16	RW	0000h
7Eh	Value Register 5 (SM1VAL5)	16	RW	0000h
80h	Fractional Control Register (SM1FRCTRL)	16	RW	0000h
82h	Output Control Register (SM1OCTRL)	16	RW	0000h
84h	Status Register (SM1STS)	16	RW	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
86h	Interrupt Enable Register (SM1INTEN)	16	RW	0000h
88h	DMA Enable Register (SM1DMAEN)	16	RW	0000h
8Ah	Output Trigger Control Register (SM1TCTRL)	16	RW	0000h
8Ch	Fault Disable Mapping Register 0 (SM1DISMAP0)	16	RW	FFFFh
90h	Deadtime Count Register 0 (SM1DTCNT0)	16	RW	07FFh
92h	Deadtime Count Register 1 (SM1DTCNT1)	16	RW	07FFh
94h	Capture Control A Register (SM1CAPTCTRLA)	16	RW	0000h
96h	Capture Compare A Register (SM1CAPTCOMPA)	16	RW	0000h
98h	Capture Control B Register (SM1CAPTCTRLB)	16	RW	0000h
9Ah	Capture Compare B Register (SM1CAPTCOMPB)	16	RW	0000h
9Ch	Capture Control X Register (SM1CAPTCTRLX)	16	RW	0000h
9Eh	Capture Compare X Register (SM1CAPTCOMPX)	16	RW	0000h
A0h	Capture Value 0 Register (SM1CVAL0)	16	R	0000h
A2h	Capture Value 0 Cycle Register (SM1CVAL0CYC)	16	R	0000h
A4h	Capture Value 1 Register (SM1CVAL1)	16	R	0000h
A6h	Capture Value 1 Cycle Register (SM1CVAL1CYC)	16	R	0000h
A8h	Capture Value 2 Register (SM1CVAL2)	16	R	0000h
AAh	Capture Value 2 Cycle Register (SM1CVAL2CYC)	16	R	0000h
ACh	Capture Value 3 Register (SM1CVAL3)	16	R	0000h
AEh	Capture Value 3 Cycle Register (SM1CVAL3CYC)	16	R	0000h
B0h	Capture Value 4 Register (SM1CVAL4)	16	R	0000h
B2h	Capture Value 4 Cycle Register (SM1CVAL4CYC)	16	R	0000h
B4h	Capture Value 5 Register (SM1CVAL5)	16	R	0000h
B6h	Capture Value 5 Cycle Register (SM1CVAL5CYC)	16	R	0000h
B8h	Phase Delay Register (SM1PHASEDLY)	16	RW	0000h
BAh	Capture PWM_A Input Filter Register (SM1CAPTFILTA)	16	RW	0000h
BCh	Capture PWM_B Input Filter Register (SM1CAPTFILTB)	16	RW	0000h
BEh	Capture PWM_X Input Filter Register (SM1CAPTFILTX)	16	RW	0000h
C0h	Counter Register (SM2CNT)	16	R	0000h
C2h	Initial Count Register (SM2INIT)	16	RW	0000h
C4h	Control 2 Register (SM2CTRL2)	16	RW	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
C6h	Control Register (SM2CTRL)	16	RW	0400h
CAh	Value Register 0 (SM2VAL0)	16	RW	0000h
CCh	Fractional Value Register 1 (SM2FRACVAL1)	16	RW	0000h
CEh	Value Register 1 (SM2VAL1)	16	RW	0000h
D0h	Fractional Value Register 2 (SM2FRACVAL2)	16	RW	0000h
D2h	Value Register 2 (SM2VAL2)	16	RW	0000h
D4h	Fractional Value Register 3 (SM2FRACVAL3)	16	RW	0000h
D6h	Value Register 3 (SM2VAL3)	16	RW	0000h
D8h	Fractional Value Register 4 (SM2FRACVAL4)	16	RW	0000h
DAh	Value Register 4 (SM2VAL4)	16	RW	0000h
DCh	Fractional Value Register 5 (SM2FRACVAL5)	16	RW	0000h
DEh	Value Register 5 (SM2VAL5)	16	RW	0000h
E0h	Fractional Control Register (SM2FRCTRL)	16	RW	0000h
E2h	Output Control Register (SM2OCTRL)	16	RW	0000h
E4h	Status Register (SM2STS)	16	RW	0000h
E6h	Interrupt Enable Register (SM2INTEN)	16	RW	0000h
E8h	DMA Enable Register (SM2DMAEN)	16	RW	0000h
EAh	Output Trigger Control Register (SM2TCTRL)	16	RW	0000h
ECh	Fault Disable Mapping Register 0 (SM2DISMAP0)	16	RW	FFFFh
F0h	Deadtime Count Register 0 (SM2DTCNT0)	16	RW	07FFh
F2h	Deadtime Count Register 1 (SM2DTCNT1)	16	RW	07FFh
F4h	Capture Control A Register (SM2CAPTCTRLA)	16	RW	0000h
F6h	Capture Compare A Register (SM2CAPTCOMPA)	16	RW	0000h
F8h	Capture Control B Register (SM2CAPTCTRLB)	16	RW	0000h
FAh	Capture Compare B Register (SM2CAPTCOMPB)	16	RW	0000h
FCh	Capture Control X Register (SM2CAPTCTRLX)	16	RW	0000h
FEh	Capture Compare X Register (SM2CAPTCOMPX)	16	RW	0000h
100h	Capture Value 0 Register (SM2CVAL0)	16	R	0000h
102h	Capture Value 0 Cycle Register (SM2CVAL0CYC)	16	R	0000h
104h	Capture Value 1 Register (SM2CVAL1)	16	R	0000h
106h	Capture Value 1 Cycle Register (SM2CVAL1CYC)	16	R	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
108h	Capture Value 2 Register (SM2CVAL2)	16	R	0000h
10Ah	Capture Value 2 Cycle Register (SM2CVAL2CYC)	16	R	0000h
10Ch	Capture Value 3 Register (SM2CVAL3)	16	R	0000h
10Eh	Capture Value 3 Cycle Register (SM2CVAL3CYC)	16	R	0000h
110h	Capture Value 4 Register (SM2CVAL4)	16	R	0000h
112h	Capture Value 4 Cycle Register (SM2CVAL4CYC)	16	R	0000h
114h	Capture Value 5 Register (SM2CVAL5)	16	R	0000h
116h	Capture Value 5 Cycle Register (SM2CVAL5CYC)	16	R	0000h
118h	Phase Delay Register (SM2PHASEDLY)	16	RW	0000h
11Ah	Capture PWM_A Input Filter Register (SM2CAPTFILTA)	16	RW	0000h
11Ch	Capture PWM_B Input Filter Register (SM2CAPTFILTB)	16	RW	0000h
11Eh	Capture PWM_X Input Filter Register (SM2CAPTFILTX)	16	RW	0000h
120h	Counter Register (SM3CNT)	16	R	0000h
122h	Initial Count Register (SM3INIT)	16	RW	0000h
124h	Control 2 Register (SM3CTRL2)	16	RW	0000h
126h	Control Register (SM3CTRL)	16	RW	0400h
12Ah	Value Register 0 (SM3VAL0)	16	RW	0000h
12Ch	Fractional Value Register 1 (SM3FRACVAL1)	16	RW	0000h
12Eh	Value Register 1 (SM3VAL1)	16	RW	0000h
130h	Fractional Value Register 2 (SM3FRACVAL2)	16	RW	0000h
132h	Value Register 2 (SM3VAL2)	16	RW	0000h
134h	Fractional Value Register 3 (SM3FRACVAL3)	16	RW	0000h
136h	Value Register 3 (SM3VAL3)	16	RW	0000h
138h	Fractional Value Register 4 (SM3FRACVAL4)	16	RW	0000h
13Ah	Value Register 4 (SM3VAL4)	16	RW	0000h
13Ch	Fractional Value Register 5 (SM3FRACVAL5)	16	RW	0000h
13Eh	Value Register 5 (SM3VAL5)	16	RW	0000h
140h	Fractional Control Register (SM3FRCTRL)	16	RW	0000h
142h	Output Control Register (SM3OCTRL)	16	RW	0000h
144h	Status Register (SM3STS)	16	RW	0000h
146h	Interrupt Enable Register (SM3INTEN)	16	RW	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
148h	DMA Enable Register (SM3DMAEN)	16	RW	0000h
14Ah	Output Trigger Control Register (SM3TCTRL)	16	RW	0000h
14Ch	Fault Disable Mapping Register 0 (SM3DISMAP0)	16	RW	FFFFh
150h	Deadtime Count Register 0 (SM3DTCNT0)	16	RW	07FFh
152h	Deadtime Count Register 1 (SM3DTCNT1)	16	RW	07FFh
154h	Capture Control A Register (SM3CAPTCTRLA)	16	RW	0000h
156h	Capture Compare A Register (SM3CAPTCOMPA)	16	RW	0000h
158h	Capture Control B Register (SM3CAPTCTRLB)	16	RW	0000h
15Ah	Capture Compare B Register (SM3CAPTCOMPB)	16	RW	0000h
15Ch	Capture Control X Register (SM3CAPTCTRLX)	16	RW	0000h
15Eh	Capture Compare X Register (SM3CAPTCOMPX)	16	RW	0000h
160h	Capture Value 0 Register (SM3CVAL0)	16	R	0000h
162h	Capture Value 0 Cycle Register (SM3CVAL0CYC)	16	R	0000h
164h	Capture Value 1 Register (SM3CVAL1)	16	R	0000h
166h	Capture Value 1 Cycle Register (SM3CVAL1CYC)	16	R	0000h
168h	Capture Value 2 Register (SM3CVAL2)	16	R	0000h
16Ah	Capture Value 2 Cycle Register (SM3CVAL2CYC)	16	R	0000h
16Ch	Capture Value 3 Register (SM3CVAL3)	16	R	0000h
16Eh	Capture Value 3 Cycle Register (SM3CVAL3CYC)	16	R	0000h
170h	Capture Value 4 Register (SM3CVAL4)	16	R	0000h
172h	Capture Value 4 Cycle Register (SM3CVAL4CYC)	16	R	0000h
174h	Capture Value 5 Register (SM3CVAL5)	16	R	0000h
176h	Capture Value 5 Cycle Register (SM3CVAL5CYC)	16	R	0000h
178h	Phase Delay Register (SM3PHASEDLY)	16	RW	0000h
17Ah	Capture PWM_A Input Filter Register (SM3CAPTFILTA)	16	RW	0000h
17Ch	Capture PWM_B Input Filter Register (SM3CAPTFILTB)	16	RW	0000h
17Eh	Capture PWM_X Input Filter Register (SM3CAPTFILTX)	16	RW	0000h
180h	Output Enable Register (OUTEN)	16	RW	0000h
182h	Mask Register (MASK)	16	RW	0000h
184h	Software Controlled Output Register (SWCOUT)	16	RW	0000h
186h	PWM Source Select Register (DTSRCSEL)	16	RW	0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
188h	Master Control Register (MCTRL)	16	RW	0000h
18Ah	Master Control 2 Register (MCTRL2)	16	RW	0000h
18Ch	Fault Control Register (FCTRL0)	16	RW	0000h
18Eh	Fault Status Register (FSTS0)	16	RW	See section
190h	Fault Filter Register (FFILT0)	16	RW	0000h
192h	Fault Test Register (FTST0)	16	RW	0000h
194h	Fault Control 2 Register (FCTRL20)	16	RW	0000h

54.5.2 Counter Register (SM0CNT - SM3CNT)

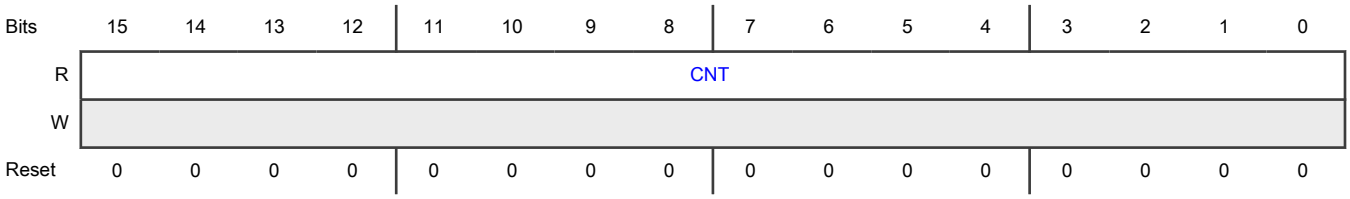
Offset

Register	Offset
SM0CNT	0h
SM1CNT	60h
SM2CNT	C0h
SM3CNT	120h

Function

This field displays the state of the signed 16-bit submodule counter. This register is not byte accessible. Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CNT	Counter Register Bits

54.5.3 Initial Count Register (SM0INIT - SM3INIT)

Offset

Register	Offset
SM0INIT	2h
SM1INIT	62h
SM2INIT	C2h
SM3INIT	122h

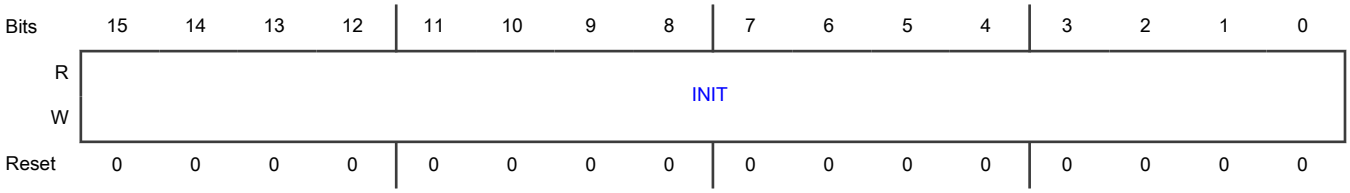
Function

The 16-bit signed value in this buffered register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of CTRL2[INIT_SEL]) or when CTRL2[FORCE] is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

The INIT register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Diagram



Fields

Field	Function
15-0 INIT	Initial Count Register Bits

54.5.4 Control 2 Register (SM0CTRL2 - SM3CTRL2)

Offset

Register	Offset
SM0CTRL2	4h
SM1CTRL2	64h
SM2CTRL2	C4h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
SM3CTRL2	124h

Function

Contains control fields for clock select and forcing output and initialization.This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DBGEN	0	INDEP	PWM23_I...	PWM45_I...	PWMX_IN...	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_...	CLK_SEL	
W										FORCE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15 DBGEN	Debug Enable When set to one, the PWM continues to run while the chip is in Debug mode. If the device enters Debug mode and this bit is zero, then the PWM outputs are disabled until Debug mode is exited. At that point, the PWM pins resume operation as programmed in the PWM registers.
14 —	Reserved
13 INDEP	Independent or Complementary Pair Operation This bit determines if the PWM_A and PWM_B channels are independent PWMs or a complementary PWM pair. 0b - PWM_A and PWM_B form a complementary PWM pair. 1b - PWM_A and PWM_B outputs are independent PWMs.
12 PWM23_INIT	PWM23 Initial Value This bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT is asserted.
11 PWM45_INIT	PWM45 Initial Value This bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT is asserted.
10 PWMX_INIT	PWM_X Initial Value This bit determines the initial value for PWM_X and the value to which it is forced when FORCE_INIT is asserted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 INIT_SEL	Initialization Control Select These bits control the source of the INIT signal which goes to the counter. 00b - Local sync (PWM_X) causes initialization. 01b - Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it forces the INIT signal to logic 0. The submodule counter will only re-initialize when a master reload occurs. 10b - Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it forces the INIT signal to logic 0. 11b - EXT_SYNC causes initialization.
7 FRCEN	Force Enable This bit allows the CTRL2[FORCE] signal to initialize the counter without regard to the signal selected by CTRL2[INIT_SEL]. This is a software controlled initialization. A forced initialization will also assert the register reload if MCTRL[LDOK] is set. 0b - Initialization from a FORCE_OUT is disabled. 1b - Initialization from a FORCE_OUT is enabled.
6 FORCE	Force Initialization If CTRL2[FORCE_SEL] is set to 000, writing a 1 to this bit results in a FORCE_OUT event. This causes the following actions to be taken: • The PWM_A and PWM_B output pins assume values based on DTSRCSEL[SMxSEL23] and DTSRCSEL[SMxSEL45]. • If CTRL2[FRCEN] is set, the counter value is initialized with the INIT register value only when the submodule MCTRL[RUN] = 1 or CTRL2[CLK_SEL] = 2.
5-3 FORCE_SEL	Force Select This bit determines the source of the FORCE OUTPUT signal for this submodule. 000b - The local force signal, CTRL2[FORCE], from this submodule is used to force updates. 001b - The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it holds the FORCE OUTPUT signal to logic 0. 010b - The local reload signal from this submodule is used to force updates without regard to the state of LDOK. 011b - The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it holds the FORCE OUTPUT signal to logic 0. 100b - The local sync signal from this submodule is used to force updates. 101b - The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it holds the FORCE OUTPUT signal to logic 0. 110b - The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111b - The external sync signal, EXT_SYNC, from outside the PWM module causes updates.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 RELOAD_SEL	Reload Source Select This bit determines the source of the RELOAD signal for this submodule. When this bit is set, MCTRL[LDOK[0]] for submodule 0 should be used since the local MCTRL[LDOK] will be ignored. 0b - The local RELOAD signal is used to reload registers. 1b - The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it forces the RELOAD signal to logic 0.
1-0 CLK_SEL	Clock Source Select These bits determine the source of the clock signal for this submodule. 00b - The IPBus clock is used as the clock for the local prescaler and counter. 01b - EXT_CLK is used as the clock for the local prescaler and counter. 10b - Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it forces the clock to logic 0. 11b - Reserved

54.5.5 Control Register (SM0CTRL - SM3CTRL)

Offset

Register	Offset
SM0CTRL	6h
SM1CTRL	66h
SM2CTRL	C6h
SM3CTRL	126h

Function
Includes control settings for timing, loading, and buffering.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LDFQ				HALF	FULL	DT		COMP MODE	PRSC			SPLIT	LDMOD	DBLX	DBLEN
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-12 LDFQ	Load Frequency These buffered bits select the PWM load frequency. Reset clears LDFQ, selecting loading every PWM opportunity. A PWM opportunity is determined by HALF and FULL. The register bits are write-protected by MCTRL2[WRPROT] bits. **NOTE** LDFQ takes effect when the current load cycle is complete, regardless of the state of MCTRL[LDOK]. Reading LDFQ reads the buffered values and not necessarily the values currently in effect. 0000b - Every PWM opportunity 0001b - Every 2 PWM opportunities 0010b - Every 3 PWM opportunities 0011b - Every 4 PWM opportunities 0100b - Every 5 PWM opportunities 0101b - Every 6 PWM opportunities 0110b - Every 7 PWM opportunities 0111b - Every 8 PWM opportunities 1000b - Every 9 PWM opportunities 1001b - Every 10 PWM opportunities 1010b - Every 11 PWM opportunities 1011b - Every 12 PWM opportunities 1100b - Every 13 PWM opportunities 1101b - Every 14 PWM opportunities 1110b - Every 15 PWM opportunities 1111b - Every 16 PWM opportunities
11 HALF	Half Cycle Reload This bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be halfway through the PWM cycle. This bit is write-protected by MCTRL2[WRPROT] bits.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Half-cycle reloads disabled. 1b - Half-cycle reloads enabled.
10 FULL	Full Cycle Reload This bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either CTRL[HALF] or CTRL[FULL] must be set to move the buffered data into the registers used by the PWM generators or CTRL[LDMOD] must be set. If both CTRL[HALF] and CTRL[FULL] are set, then reloads can occur twice per cycle. This bit is write-protected by MCTRL2[WRPROT] bits. 0b - Full-cycle reloads disabled. 1b - Full-cycle reloads enabled.
9-8 DT	Deadtime These bits reflect the sampled values of the PWM_X input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits. The register bits are write-protected by MCTRL2[WRPROT] bits.
7 COMPMODE	Compare Mode This bit controls how comparisons are made between the VAL* registers and the PWM submodule counter. This bit can be written one time after which it requires a reset to release the bit for writing again. 0b - The VAL* registers and the PWM counter are compared using an "equal to" method. This means that PWM edges are only produced when the counter is equal to one of the VAL* register values. This implies that a PWM_A output that is high at the end of a period maintains this state until a match with VAL3 clears the output in the following period. 1b - The VAL* registers and the PWM counter are compared using an "equal to or greater than" method. This means that PWM edges are produced when the counter is equal to or greater than one of the VAL* register values. This implies that a PWM_A output that is high at the end of a period could go low at the start of the next period if the starting counter value is greater than (but not necessarily equal to) the new VAL3 value.
6-4 PRSC	Prescaler These buffered bits select the divide ratio of the PWM clock frequency selected by CTRL2[CLK_SEL]. **NOTE** Reading CTRL[PRSC] reads the buffered values and not necessarily the values currently in effect. CTRL[PRSC] takes effect at the beginning of the next PWM cycle and only when the load okay bit MCTRL[LDOK] is set, or CTRL[LDMOD] is set. This field cannot be written when MCTRL[LDOK] is set. 000b - Prescaler 1. PWM clock frequency = f_{clk} 001b - Prescaler 2. PWM clock frequency = f_{clk} /2 010b - Prescaler 4. PWM clock frequency = f_{clk} /4 011b - Prescaler 8. PWM clock frequency = f_{clk} /8

Table continues on the next page...

Table continued from the previous page...

Field	Function
	100b - Prescaler 16. PWM clock frequency = f_{clk} /16 101b - Prescaler 32. PWM clock frequency = f_{clk} /32 110b - Prescaler 64. PWM clock frequency = f_{clk} /64 111b - Prescaler 128. PWM clock frequency = f_{clk} /128
3 SPLIT	Split the DBLPWM signal to PWM_A and PWM_B This bit is used in independent mode when DBLEN is set. This bit allows the two PWM pulses generated by DBLEN to be split with one pulse on PWM_A and one on PWM_B. The two pulses within the same PWM period are created by an XOR function of the PWM_A and PWM_B sources. The splitting function causes PWM_A to output the pulse that occurs when the PWM_A source is 1 and the PWM_B source is 0. The PWM_B output occurs when the PWM_B source is 1 and the PWM_A source is 0. (See Double switching PWMs .) This bit is write-protected by MCTRL2[WRPROT] bits. 0b - DBLPWM is not split. PWM_A and PWM_B each have double pulses. 1b - DBLPWM is split to PWM_A and PWM_B.
2 LDMOD	Load Mode Select This bit selects the timing of loading the buffered registers for this submodule. This bit is write-protected by MCTRL2[WRPROT] bits. 0b - Buffered registers of this submodule are loaded and take effect at the next PWM reload if MCTRL[LDOK] is set. 1b - Buffered registers of this submodule are loaded and take effect immediately upon MCTRL[LDOK] being set. In this case, it is not necessary to set CTRL[FULL] or CTRL[HALF].
1 DBLX	PWM_X Double Switching Enable This bit enables the double switching behavior on PWM_X. When this bit is set, the PWM_X output shall be the exclusive OR combination of PWM_A and PWM_B prior to polarity and masking considerations. This bit is write-protected by MCTRL2[WRPROT] bits. 0b - PWM_X double pulse disabled. 1b - PWM_X double pulse enabled.
0 DBLEN	Double Switching Enable This bit enables the double switching PWM behavior (See Double switching PWMs). This bit is write-protected by MCTRL2[WRPROT] bits. 0b - Double switching disabled. 1b - Double switching enabled.

54.5.6 Value Register 0 (SM0VAL0 - SM3VAL0)

Offset

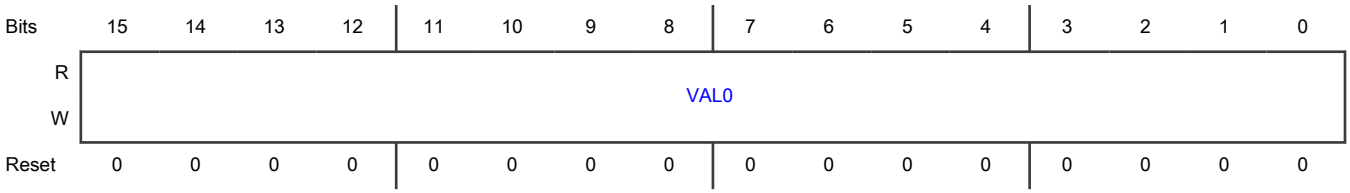
Register	Offset
SM0VAL0	Ah
SM1VAL0	6Ah
SM2VAL0	CAh
SM3VAL0	12Ah

Function

NOTE

The VAL0 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL0 cannot be written when MCTRL[LDOK] is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value that the PWM generator is currently using.

Diagram



Fields

Field	Function
15-0 VAL0	Value 0 The 16-bit signed value in this buffered register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWM_X signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE** The actual behavior takes effect when counter equal VAL0+1.

54.5.7 Fractional Value Register 1 (SM0FRACVAL1 - SM3FRACVAL1)

Offset

Register	Offset
SM0FRACVAL1	Ch
SM1FRACVAL1	6Ch

Table continues on the next page...

Table continued from the previous page...

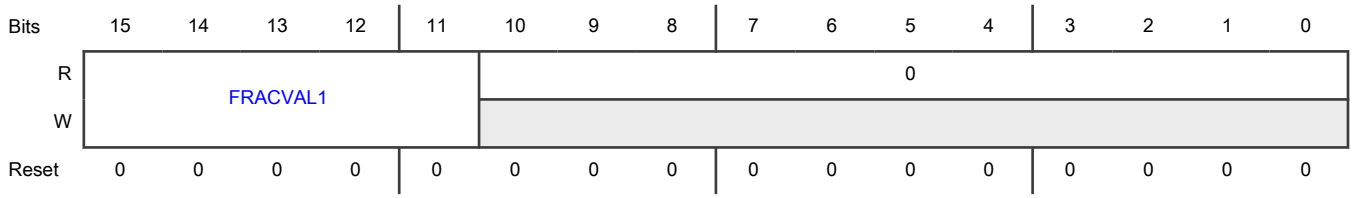
Register	Offset
SM2FRACVAL1	CCh
SM3FRACVAL1	12Ch

Function

NOTE

The FRACVAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL1 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Diagram



Fields

Field	Function
15-11 FRACVAL1	Fractional Value 1 These bits act as a fractional addition to the value in the VAL1 register which controls the PWM period. With fractional delay enabled, PWM works in digital dithering mode. The PWM period is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL1 is temporarily incremented, and the PWM period is extended by one clock to compensate for the accumulated fractional values.
10-0 —	Reserved

54.5.8 Value Register 1 (SM0VAL1 - SM3VAL1)

Offset

Register	Offset
SM0VAL1	Eh
SM1VAL1	6Eh
SM2VAL1	CEh
SM3VAL1	12Eh

Function

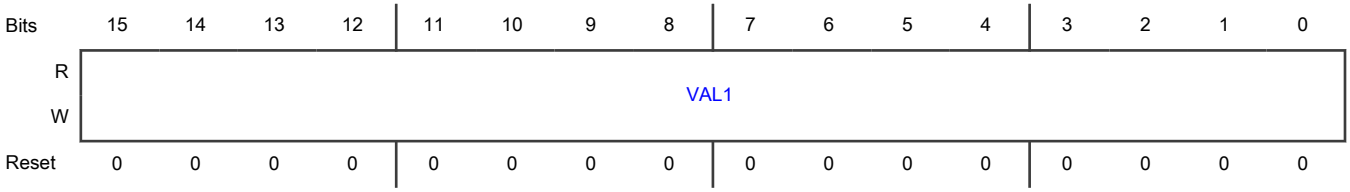
NOTE

The VAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL1 cannot be written when MCTRL[LDOK] is set. Reading VAL1 reads the value in a buffer. It is not necessarily the value that the PWM generator is currently using.

When using FRACVAL1, limit the maximum value of VAL1 to 0xFFFE for unsigned applications or to 0x7FFE for signed applications, to avoid counter rollovers caused by accumulating the fractional period defined by FRACVAL1.

If the VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWM_X output. After the count reaches VAL1, the PWM_X output is low for a minimum of one count every cycle. When the Master Sync signal (only originated by the Local Sync from submodule 0) is used to control the timer period, the VAL1 register can be free for other functions such as PWM generation without the duty cycle limitation.

Diagram



Fields

Field	Function
15-0 VAL1	Value 1 The 16-bit signed value written to this buffered register defines the modulo count value (maximum count) for the submodule counter. When reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local sync signal while resetting PWM_X. This register is not byte accessible. **NOTE** The actual behavior takes effect when the counter equals VAL1+1.

54.5.9 Fractional Value Register 2 (SM0FRACVAL2 - SM3FRACVAL2)

Offset

Register	Offset
SM0FRACVAL2	10h
SM1FRACVAL2	70h
SM2FRACVAL2	D0h
SM3FRACVAL2	130h

Function

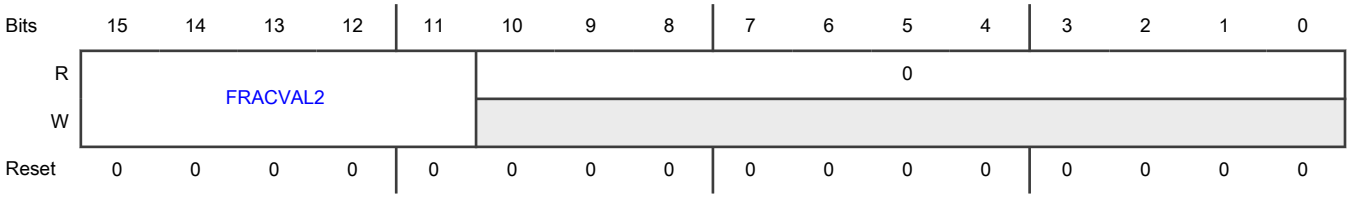
NOTE

The FRACVAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL2 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

NOTE

FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.

Diagram



Fields

Field	Function
15-11 FRACVAL2	Fractional Value 2 These bits act as a fractional addition to the value in the VAL2 register which controls the PWM_A turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. With fractional delay enabled, PWM works in digital dithering mode. The PWM_A turn on delay is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL2 is temporarily incremented, and the PWM_A turn on delay is extended by one clock to compensate for the accumulated fractional values.
10-0 —	Reserved

54.5.10 Value Register 2 (SM0VAL2 - SM3VAL2)

Offset

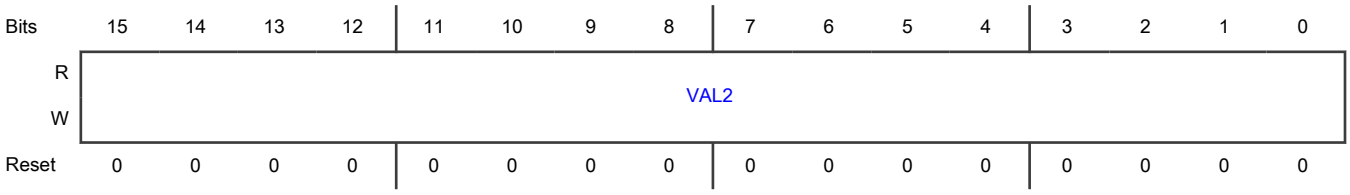
Register	Offset
SM0VAL2	12h
SM1VAL2	72h
SM2VAL2	D2h
SM3VAL2	132h

Function

NOTE

The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL2 cannot be written when MCTRL[LDOK] is set. Reading VAL2 reads the value in a buffer and not necessarily the value that the PWM generator is currently using.

Diagram



Fields

Field	Function
15-0 VAL2	Value 2 The 16-bit signed value in this buffered register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE** The actual behavior takes effect when the counter equals VAL2+1.

54.5.11 Fractional Value Register 3 (SM0FRACVAL3 - SM3FRACVAL3)

Offset

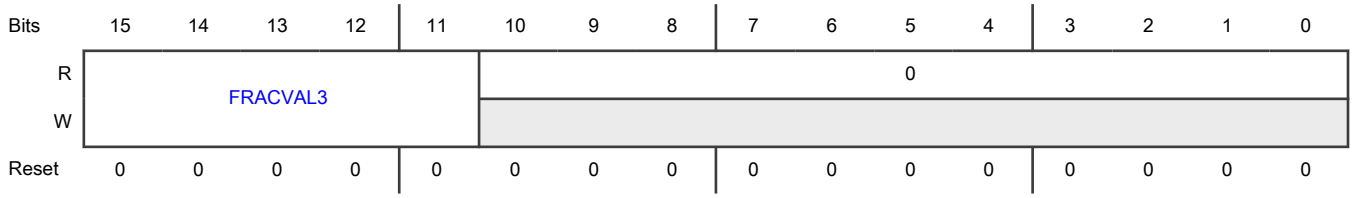
Register	Offset
SM0FRACVAL3	14h
SM1FRACVAL3	74h
SM2FRACVAL3	D4h
SM3FRACVAL3	134h

Function

NOTE

The FRACVAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL3 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Diagram



Fields

Field	Function
15-11 FRACVAL3	Fractional Value 3 These bits act as a fractional addition to the value in the VAL3 register which controls the PWM_A turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less. With fractional delay enabled, PWM works in digital dithering mode. The PWM_A turn off delay is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL3 is temporarily incremented, and the PWM_A turn off delay is extended by one clock to compensate for the accumulated fractional values.
10-0 —	Reserved

54.5.12 Value Register 3 (SM0VAL3 - SM3VAL3)

Offset

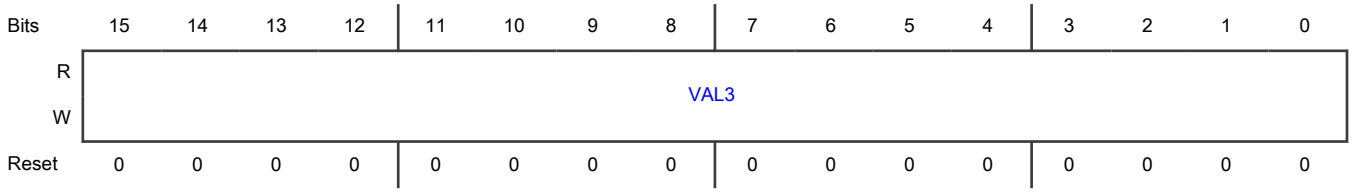
Register	Offset
SM0VAL3	16h
SM1VAL3	76h
SM2VAL3	D6h
SM3VAL3	136h

Function

NOTE

The VAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL3 cannot be written when MCTRL[LDOK] is set. Reading VAL3 reads the value in a buffer and not necessarily the value that the PWM generator is currently using.

Diagram



Fields

Field	Function
15-0 VAL3	Value 3 The 16-bit signed value in this buffered register defines the count value to set PWM23 low. This register is not byte accessible. **NOTE** The actual behavior takes effect when the counter equals VAL3+1.

54.5.13 Fractional Value Register 4 (SM0FRACVAL4 - SM3FRACVAL4)

Offset

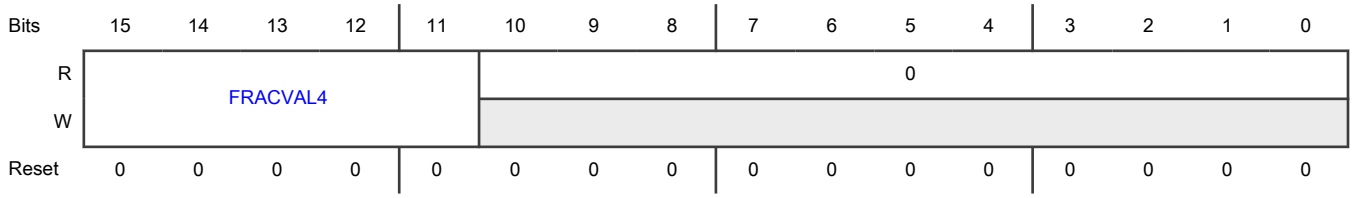
Register	Offset
SM0FRACVAL4	18h
SM1FRACVAL4	78h
SM2FRACVAL4	D8h
SM3FRACVAL4	138h

Function

NOTE

The FRACVAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL4 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Diagram



Fields

Field	Function
15-11 FRACVAL4	Fractional Value 4 These bits act as a fractional addition to the value in the VAL4 register which controls the PWM_B turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less. With fractional delay enabled, PWM works in digital dithering mode. The PWM_B turn on delay is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL4 is temporarily incremented, and the PWM_B turn on delay is extended by one clock to compensate for the accumulated fractional values.
10-0 —	Reserved

54.5.14 Value Register 4 (SM0VAL4 - SM3VAL4)

Offset

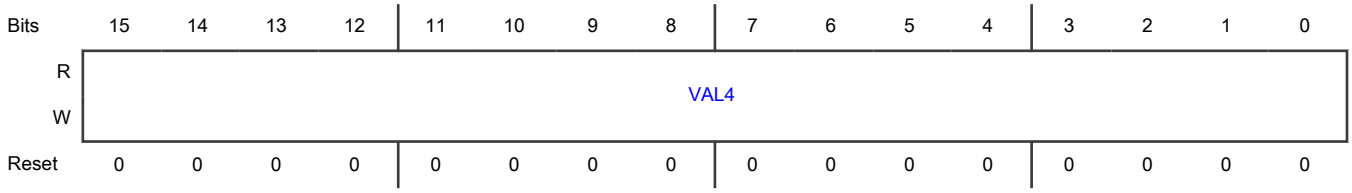
Register	Offset
SM0VAL4	1Ah
SM1VAL4	7Ah
SM2VAL4	DAh
SM3VAL4	13Ah

Function

NOTE

The VAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL4 cannot be written when MCTRL[LDOK] is set. Reading VAL4 reads the value in a buffer and not necessarily the value that the PWM generator is currently using.

Diagram



Fields

Field	Function
15-0 VAL4	Value 4 The 16-bit signed value in this buffered register defines the count value to set PWM45 high. This register is not byte accessible. **NOTE** The actual behavior takes effect when the counter equals VAL4+1.

54.5.15 Fractional Value Register 5 (SM0FRACVAL5 - SM3FRACVAL5)

Offset

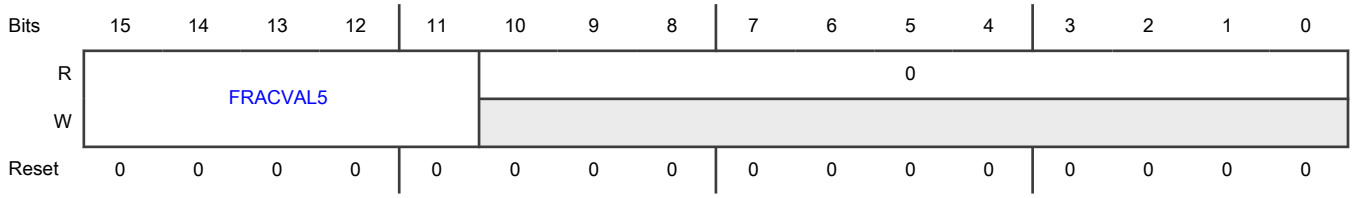
Register	Offset
SM0FRACVAL5	1Ch
SM1FRACVAL5	7Ch
SM2FRACVAL5	DCh
SM3FRACVAL5	13Ch

Function

NOTE

The FRACVAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL5 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Diagram



Fields

Field	Function
15-11 FRACVAL5	Fractional Value 5 These bits act as a fractional addition to the value in the VAL5 register which controls the PWM_B turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less. With fractional delay enabled, PWM works in digital dithering mode. The PWM_B turn off delay is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL5 is temporarily incremented, and the PWM_B turn off delay is extended by one clock to compensate for the accumulated fractional values.
10-0 —	Reserved

54.5.16 Value Register 5 (SM0VAL5 - SM3VAL5)

Offset

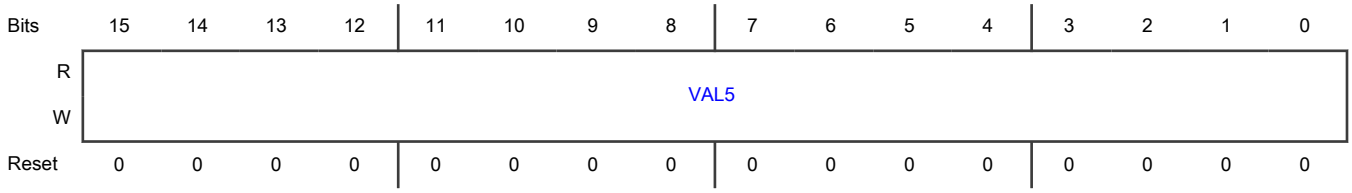
Register	Offset
SM0VAL5	1Eh
SM1VAL5	7Eh
SM2VAL5	DEh
SM3VAL5	13Eh

Function

NOTE

The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL5 cannot be written when MCTRL[LDOK] is set. Reading VAL5 reads the value in a buffer and not necessarily the value that the PWM generator is currently using.

Diagram



Fields

Field	Function
15-0 VAL5	Value 5 The 16-bit signed value in this buffered register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE** The actual behavior takes effect when the counter equals VAL5+1.

54.5.17 Fractional Control Register (SM0FRCTRL - SM3FRCTRL)

Offset

Register	Offset
SM0FRCTRL	20h
SM1FRCTRL	80h
SM2FRCTRL	E0h
SM3FRCTRL	140h

Function

NOTE

The FRAC1_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC1_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC1_EN reads the value in a buffer and not necessarily the value that the PWM generator is currently using.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEST	0							0			FRAC45_...	0	FRAC23_...	FRAC1_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15 TEST	Test Status Bit This is a test bit for factory use. This bit resets to 0 but may be either 0 or 1 during PWM operation.
14-8 —	Reserved
7-5 —	Reserved
4 FRAC45_EN	Fractional Cycle Placement Enable for PWM_B This bit is used to enable the fractional cycle edge placement of PWM_B using the FRACVAL4 and FRACVAL5 registers. When disabled, bypass the fractional cycle edge placement of PWM_B. **NOTE** The FRAC45_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC45_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC45_EN reads the value in a buffer and not necessarily the value that the PWM generator is currently using. 0b - Disable fractional cycle placement for PWM_B. 1b - Enable fractional cycle placement for PWM_B.
3 —	Reserved
2 FRAC23_EN	Fractional Cycle Placement Enable for PWM_A This bit is used to enable the fractional cycle edge placement of PWM_A using the FRACVAL2 and FRACVAL3 registers. When disabled, bypass the fractional cycle edge placement of PWM_A. **NOTE** The FRAC23_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC23_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC23_EN reads the value in a buffer and not necessarily the value that the PWM generator is currently using. 0b - Disable fractional cycle placement for PWM_A. 1b - Enable fractional cycle placement for PWM_A.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 FRAC1_EN	Fractional Cycle PWM Period Enable This bit is used to enable the fractional cycle length of the PWM period using the FRACVAL1 register. When disabled, bypass the fractional cycle length of the PWM period. **NOTE** The FRAC1_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC1_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC1_EN reads the value in a buffer and not necessarily the value that the PWM generator is currently using. 0b - Disable fractional cycle length for the PWM period. 1b - Enable fractional cycle length for the PWM period.
0 —	Reserved

54.5.18 Output Control Register (SM0OCTRL - SM3OCTRL)

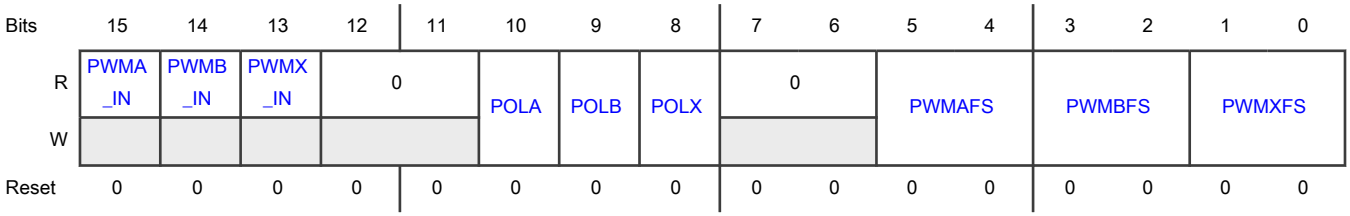
Offset

Register	Offset
SM0OCTRL	22h
SM1OCTRL	82h
SM2OCTRL	E2h
SM3OCTRL	142h

Function

Contains output controls for fault states. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15 PWMA_IN	PWM_A Input This bit shows the logic value currently being driven into the PWM_A input. The reset state is undefined.
14 PWMB_IN	PWM_B Input This bit shows the logic value currently being driven into the PWM_B input. The reset state is undefined.
13 PWMX_IN	PWM_X Input This bit shows the logic value currently being driven into the PWM_X input. The reset state is undefined.
12-11 —	Reserved
10 POLA	PWM_A Output Polarity This bit inverts the PWM_A output polarity. 0b - PWM_A output not inverted. A high level on the PWM_A pin represents the "on" or "active" state. 1b - PWM_A output inverted. A low level on the PWM_A pin represents the "on" or "active" state.
9 POLB	PWM_B Output Polarity This bit inverts the PWM_B output polarity. 0b - PWM_B output not inverted. A high level on the PWM_B pin represents the "on" or "active" state. 1b - PWM_B output inverted. A low level on the PWM_B pin represents the "on" or "active" state.
8 POLX	PWM_X Output Polarity This bit inverts the PWM_X output polarity. 0b - PWM_X output not inverted. A high level on the PWM_X pin represents the "on" or "active" state. 1b - PWM_X output inverted. A low level on the PWM_X pin represents the "on" or "active" state.
7-6 —	Reserved
5-4	PWM_A Fault State

Table continues on the next page...

Table continued from the previous page...

Field	Function
PWMAFS	These bits determine the fault state for the PWM_A output during fault conditions and Deep Sleep mode. It may also define the output state during Debug mode depending on the setting of CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10b,11b - Output is put in a high-impedance state.
3-2 PWMBFS	PWM_B Fault State These bits determine the fault state for the PWM_B output during fault conditions and Deep Sleep mode. It may also define the output state during Debug mode depending on the setting of CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10b,11b - Output is put in a high-impedance state.
1-0 PWMXFS	PWM_X Fault State These bits determine the fault state for the PWM_X output during fault conditions and Deep Sleep mode. It may also define the output state during Debug mode depending on the setting of CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10b,11b - Output is put in a high-impedance state.

54.5.19 Status Register (SM0STS - SM3STS)

Offset

Register	Offset
SM0STS	24h
SM1STS	84h
SM2STS	E4h
SM3STS	144h

Function
Contains Compare and Capture flag status.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	CFX1	CFX0	CMPF					
W			W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15 —	Reserved
14 RUF	Registers Updated Flag This bit is set when one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers has been written, which indicates potentially non-coherent data in the set of double buffered registers. Clear this bit by a proper reload sequence consisting of a reload signal while MCTRL[LDOK] = 1. Reset clears this bit. 0b - No register update has occurred since last reload. 1b - At least one of the double buffered registers has been updated since the last reload.
13 REF	Reload Error Flag This bit is set when a reload cycle occurs while MCTRL[LDOK] is 0 and the double buffered registers are in a non-coherent state (STS[RUF] = 1). Clear this bit by writing a logic one to this location. Reset clears this bit. 0b - No reload error occurred. 1b - Reload signal occurred with non-coherent data and MCTRL[LDOK] = 0.
12 RF	Reload Flag This bit is set at the beginning of every reload cycle regardless of the state of MCTRL[LDOK]. Clear this bit by writing a logic one to this location when DMAEN[VALDE] is clear (non-DMA mode). This bit can also be cleared by the DMA done signal when DMAEN[VALDE] is set (DMA mode) . Reset clears this bit. 0b - No new reload cycle since last STS[RF] clearing 1b - New reload cycle since last STS[RF] clearing
11 CFA1	Capture Flag A1 This bit is set when a capture event occurs on the Capture A1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA1DE] is set (DMA mode) . Reset clears this bit.
10 CFA0	Capture Flag A0 This bit is set when a capture event occurs on the Capture A0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA0DE] is set (DMA mode) . Reset clears this bit.
9	Capture Flag B1

Table continues on the next page...

Table continued from the previous page...

Field	Function
CFB1	This bit is set when a capture event occurs on the Capture B1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB1DE] is set (DMA mode) . Reset clears this bit.
8 CFB0	Capture Flag B0 This bit is set when a capture event occurs on the Capture B0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB0DE] is set (DMA mode) . Reset clears this bit.
7 CFX1	Capture Flag X1 This bit is set when a capture event occurs on the Capture X1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX1DE] is set (DMA mode) . Reset clears this bit.
6 CFX0	Capture Flag X0 This bit is set when a capture event occurs on the Capture X0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX0DE] is set (DMA mode) . Reset clears this bit.
5-0 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 00_0000b - No compare event has occurred for a particular VALx value. 00_0001b - A compare event has occurred for a particular VALx value.

54.5.20 Interrupt Enable Register (SM0INTEN - SM3INTEN)

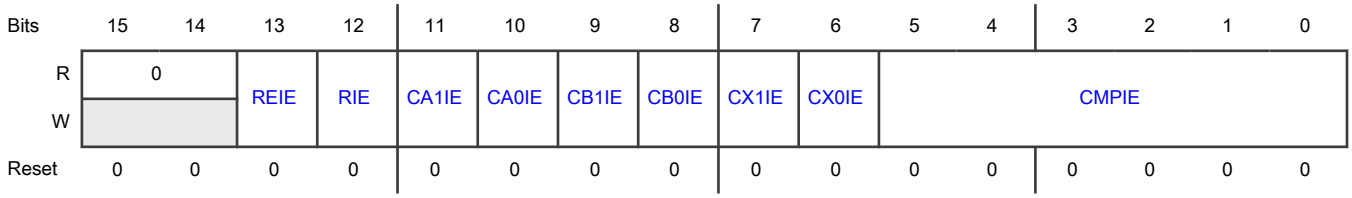
Offset

Register	Offset
SM0INTEN	26h
SM1INTEN	86h
SM2INTEN	E6h
SM3INTEN	146h

Function

Contains Compare and Capture interrupt enables.

Diagram



Fields

Field	Function
15-14 —	Reserved
13 REIE	Reload Error Interrupt Enable This bit enables the reload error flag, STS[REF], to generate CPU interrupt requests. Reset clears this bit. 0b - STS[REF] CPU interrupt requests disabled 1b - STS[REF] CPU interrupt requests enabled
12 RIE	Reload Interrupt Enable This bit enables the reload flag, STS[RF], to generate CPU interrupt requests. Reset clears this bit. 0b - STS[RF] CPU interrupt requests disabled 1b - STS[RF] CPU interrupt requests enabled
11 CA1IE	Capture A 1 Interrupt Enable This bit allows the STS[CFA1] flag to create an interrupt request to the CPU. Do not set this bit and DMAEN[CA1DE]. 0b - Interrupt request disabled for STS[CFA1] 1b - Interrupt request enabled for STS[CFA1]
10 CA0IE	Capture A 0 Interrupt Enable This bit allows the STS[CFA0] flag to create an interrupt request to the CPU. Do not set this bit and DMAEN[CA0DE]. 0b - Interrupt request disabled for STS[CFA0]. 1b - Interrupt request enabled for STS[CFA0].
9 CB1IE	Capture B 1 Interrupt Enable This bit allows the STS[CFB1] flag to create an interrupt request to the CPU. Do not set this bit and DMAEN[CB1DE]. 0b - Interrupt request disabled for STS[CFB1]. 1b - Interrupt request enabled for STS[CFB1].
8	Capture B 0 Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
CB0IE	This bit allows the STS[CFB0] flag to create an interrupt request to the CPU. Do not set this bit and DMAEN[CB0DE]. 0b - Interrupt request disabled for STS[CFB0]. 1b - Interrupt request enabled for STS[CFB0].
7 CX1IE	Capture X 1 Interrupt Enable This bit allows the STS[CFX1] flag to create an interrupt request to the CPU. Do not set this bit and DMAEN[CX1DE]. 0b - Interrupt request disabled for STS[CFX1]. 1b - Interrupt request enabled for STS[CFX1].
6 CX0IE	Capture X 0 Interrupt Enable This bit allows the STS[CFX0] flag to create an interrupt request to the CPU. Do not set this bit and DMAEN[CX0DE]. 0b - Interrupt request disabled for STS[CFX0]. 1b - Interrupt request enabled for STS[CFX0].
5-0 CMPIE	Compare Interrupt Enables These bits enable the STS[CMPF] flags to cause a compare interrupt request to the CPU. 00_0000b - The corresponding STS[CMPF] bit will not cause an interrupt request. 00_0001b - The corresponding STS[CMPF] bit will cause an interrupt request.

54.5.21 DMA Enable Register (SM0DMAEN - SM3DMAEN)

Offset

Register	Offset
SM0DMAEN	28h
SM1DMAEN	88h
SM2DMAEN	E8h
SM3DMAEN	148h

Function

Contains controls for DMA. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						VALDE	FAND	CAPTDE		CA1DE	CA0DE	CB1DE	CB0DE	CX1DE	CX0DE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-10 —	Reserved
9 VALDE	Value Registers DMA Enable This bit enables DMA write requests for the VALx and FRACVALx registers when STS[RF] is set. Reset clears this bit. 0b - DMA write requests disabled 1b - Enabled. DMA write requests for the VALx and FRACVALx registers enabled
8 FAND	FIFO Watermark AND Control This bit works in conjunction with the DMAEN[CAPTDE] field when it is set to watermark mode (DMAEN[CAPTDE] = 01). While DMAEN[CAxDE], DMAEN[CBxDE], and DMAEN[CXxDE] determine which FIFO watermarks the DMA read request is sensitive to. This bit determines if the selected watermarks are AND'ed together or OR'ed together to create the request. 0b - Selected FIFO watermarks are OR'ed together. 1b - Selected FIFO watermarks are AND'ed together.
7-6 CAPTDE	Capture DMA Enable Source Select These bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00b - Read DMA requests disabled. 01b - Exceeding a FIFO watermark sets the DMA read request. This requires at least one of DMAEN[CA1DE], DMAEN[CA0DE], DMAEN[CB1DE], DMAEN[CB0DE], DMAEN[CX1DE], or DMAEN[CX0DE] to be set to determine which watermark(s) the DMA request is sensitive. 10b - A local synchronization (VAL1 matches counter) sets the read DMA request. 11b - A local reload (STS[RF] being set) sets the read DMA request.
5 CA1DE	Capture A1 FIFO DMA Enable This bit enables DMA read requests for the Capture A1 FIFO data when STS[CFA1] is set. Reset clears this bit. Do not set this bit and INTEN[CA1IE].
4 CA0DE	Capture A0 FIFO DMA Enable This bit enables DMA read requests for the Capture A0 FIFO data when STS[CFA0] is set. Reset clears this bit. Do not set both this bit and INTEN[CA0IE].

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CB1DE	Capture B1 FIFO DMA Enable This bit enables DMA read requests for the Capture B1 FIFO data when STS[CFB1] is set. Reset clears this bit. Do not set both this bit and INTEN[CB1IE].
2 CB0DE	Capture B0 FIFO DMA Enable This bit enables DMA read requests for the Capture B0 FIFO data when STS[CFB0] is set. Reset clears this bit. Do not set both this bit and INTEN[CB0IE].
1 CX1DE	Capture X1 FIFO DMA Enable This bit enables DMA read requests for the Capture X1 FIFO data when STS[CFX1] is set. Reset clears this bit. Do not set both this bit and INTEN[CX1IE].
0 CX0DE	Capture X0 FIFO DMA Enable This bit enables DMA read requests for the Capture X0 FIFO data when STS[CFX0] is set. Reset clears this bit. Do not set both this bit and INTEN[CX0IE].

54.5.22 Output Trigger Control Register (SM0TCTRL - SM3TCTRL)

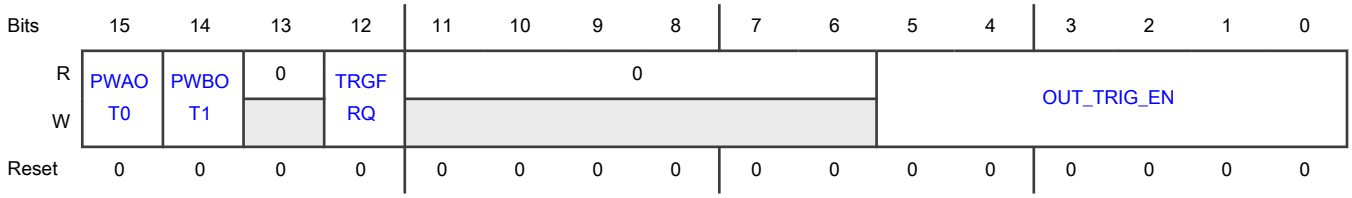
Offset

Register	Offset
SM0TCTRL	2Ah
SM1TCTRL	8Ah
SM2TCTRL	EAh
SM3TCTRL	14Ah

Function

Contains trigger controls.This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15 PWAOT0	Mux Output Trigger 0 Source Select This bit selects which signal to bring out on the PWM's PWM_MUX_TRIG0 port. 0b - Route the PWM_OUT_TRIG0 signal to PWM_MUX_TRIG0 port. 1b - Route the PWM_A output to the PWM_MUX_TRIG0 port.
14 PWBOT1	Mux Output Trigger 1 Source Select This bit selects which signal to bring out on the PWM's PWM_MUX_TRIG1 port. 0b - Route the PWM_OUT_TRIG1 signal to PWM_MUX_TRIG1 port. 1b - Route the PWM_B output to the PWM_MUX_TRIG1 port.
13 —	Reserved
12 TRGFRQ	Trigger Frequency This bit allows control over the frequency of the trigger outputs when using non-zero values of CTRL[LDFQ]. 0b - Trigger outputs are generated during every PWM period even if the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero. 1b - Trigger outputs are generated only during the final PWM period prior to a reload opportunity when the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero.
11-6 —	Reserved
5-0 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of PWM_OUT_TRIG0 and PWM_OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. **NOTE** Due to delays in creating the PWM outputs, the output trigger signals lead the PWM output edges by 2-3 clock cycles depending on the fractional cycle value being used. 1x_xxxxb - PWM_OUT_TRIG1 will set when the counter value matches the VAL5 value. x1_xxxxb - PWM_OUT_TRIG0 will set when the counter value matches the VAL4 value.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	xx_1xxxb - PWM_OUT_TRIG1 will set when the counter value matches the VAL3 value. xx_x1xxb - PWM_OUT_TRIG0 will set when the counter value matches the VAL2 value. xx_xx1xb - PWM_OUT_TRIG1 will set when the counter value matches the VAL1 value. xx_xxx1b - PWM_OUT_TRIG0 will set when the counter value matches the VAL0 value.

54.5.23 Fault Disable Mapping Register 0 (SM0DISMAP0 - SM3DISMAP0)

Offset

Register	Offset
SM0DISMAP0	2Ch
SM1DISMAP0	8Ch
SM2DISMAP0	ECh
SM3DISMAP0	14Ch

Function

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1				DIS0X				DIS0B				DIS0A			
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
15-12 —	Reserved
11-8 DIS0X	PWM_X Fault Disable Mask 0 Each of the four bits of this field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_X output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. DIS0X[0] is associated with FAULT0. DIS0X[1] is associated with FAULT1. DIS0X[2] is associated with FAULT2. DIS0X[3] is associated with FAULT3.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	A reset sets all bits in this field.
7-4 DIS0B	PWM_B Fault Disable Mask 0 Each of the four bits of this field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_B output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. DIS0B[0] is associated with FAULT0. DIS0B[1] is associated with FAULT1. DIS0B[2] is associated with FAULT2. DIS0B[3] is associated with FAULT3. A reset sets all bits in this field.
3-0 DIS0A	PWM_A Fault Disable Mask 0 Each of the four bits of this field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_A output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. DIS0A[0] is associated with FAULT0. DIS0A[1] is associated with FAULT1. DIS0A[2] is associated with FAULT2. DIS0A[3] is associated with FAULT3. A reset sets all bits in this field.

54.5.24 Deadtime Count Register 0 (SM0DTCNT0 - SM3DTCNT0)

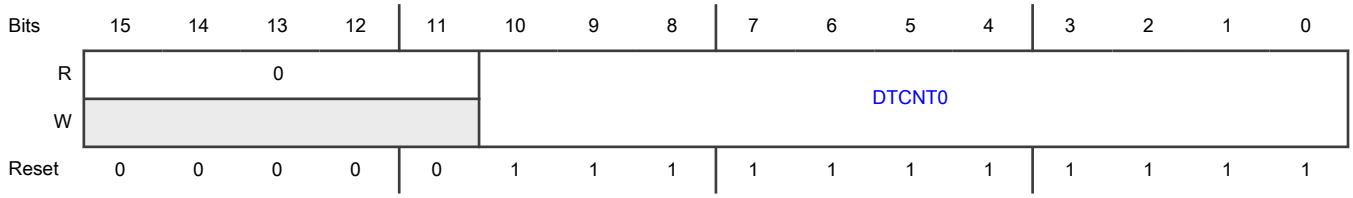
Offset

Register	Offset
SM0DTCNT0	30h
SM1DTCNT0	90h
SM2DTCNT0	F0h
SM3DTCNT0	150h

Function

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles, when fractional delay is not enabled. The DTCNTx registers are not byte accessible.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-0 DTCNT0	Deadtime Count Register 0 This field is used to control the deadtime during 0 to 1 transitions of the PWM_A output (assuming normal polarity).

54.5.25 Deadtime Count Register 1 (SM0DTCNT1 - SM3DTCNT1)

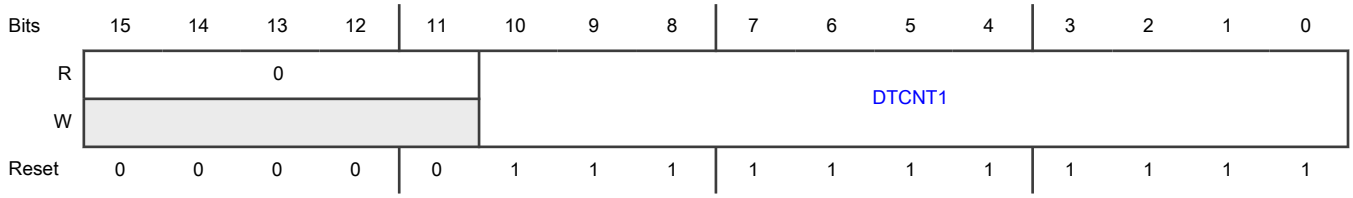
Offset

Register	Offset
SM0DTCNT1	32h
SM1DTCNT1	92h
SM2DTCNT1	F2h
SM3DTCNT1	152h

Function

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles, when fractional delay is not enabled. The DTCNTx registers are not byte accessible.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-0 DTCNT1	Deadtime Count Register 1 This field is used to control the deadtime during 0 to 1 transitions of the complementary PWM_B output.

54.5.26 Capture Control A Register (SM0CAPTCTRLA)

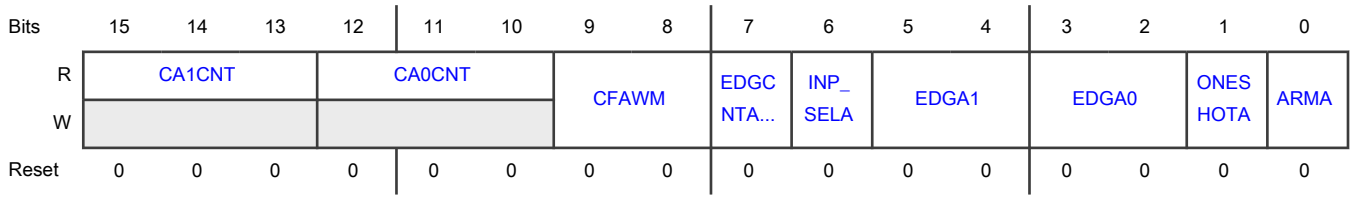
Offset

Register	Offset
SM0CAPTCTRLA	34h

Function

Contains capture controls for mode A. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1.)
12-10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1.)

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTA_EN	Edge Counter A Enable This field enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELA	Input Select A This field selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_A input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields to enable one or both of the capture registers.
5-4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is

Table continues on the next page...

Table continued from the previous page...

Field	Function
	disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLA[ARMA] is cleared. No further captures are performed until CAPTCTRLA[ARMA] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

54.5.27 Capture Compare A Register (SM0CAPTCOMPA)

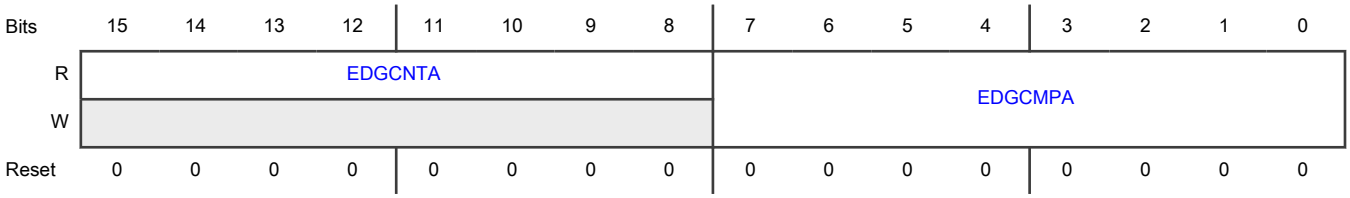
Offset

Register	Offset
SM0CAPTCOMPA	36h

Function

Contains capture and compare values for mode A.This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTA	Edge Counter A This field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

54.5.28 Capture Control B Register (SM0CAPTCTRLB)

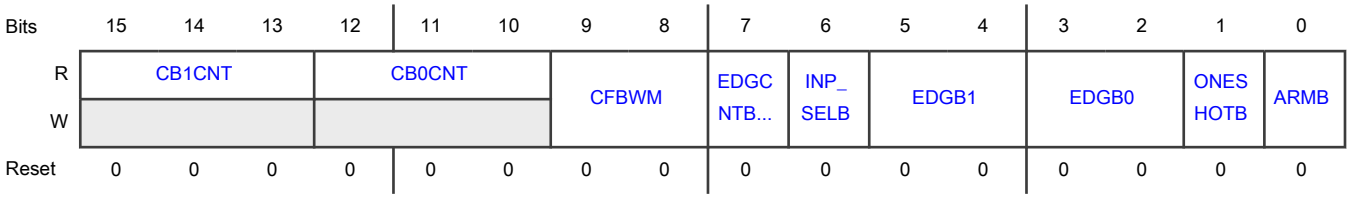
Offset

Register	Offset
SM0CAPTCTRLB	38h

Function

Contains capture controls for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1.)
12-10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1.)
9-8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTB_EN	Edge Counter B Enable This field enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELB	Input Select B This field selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_B input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the

Table continues on the next page...

Table continued from the previous page...

Field	Function
	CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields to enable one or both of the capture registers.
5-4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLB[ARMB] is cleared. No further captures are performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

54.5.29 Capture Compare B Register (SM0CAPTCOMPB)

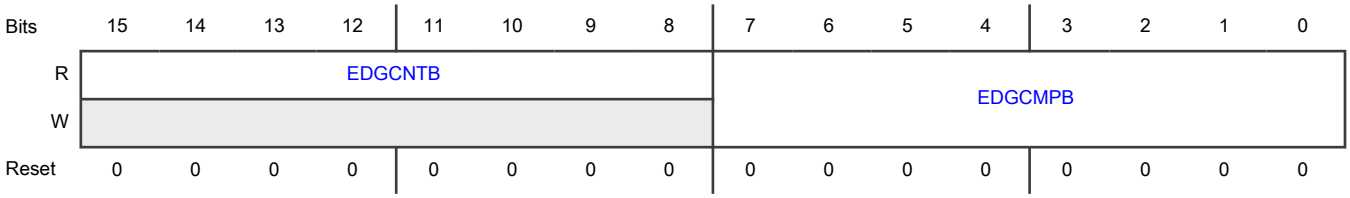
Offset

Register	Offset
SM0CAPTCOMPB	3Ah

Function

Contains capture and compare values for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTB	Edge Counter B This field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

54.5.30 Capture Control X Register (SM0CAPTCTRLX)

Offset

Register	Offset
SM0CAPTCTRLX	3Ch

Function

Contains capture controls for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CX1CNT				CX0CNT				EDGCNTX...	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
W							CFXWM									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1.)
12-10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1.)
9-8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_X input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields to enable one or both of the capture registers.
5-4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2	Edge X 0

Table continues on the next page...

Table continued from the previous page...

Field	Function
EDGX0	These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTX	One Shot Mode Aux This field selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and the ARMX bit is cleared. No further captures are performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

54.5.31 Capture Compare X Register (SM0CAPTCOMPX)

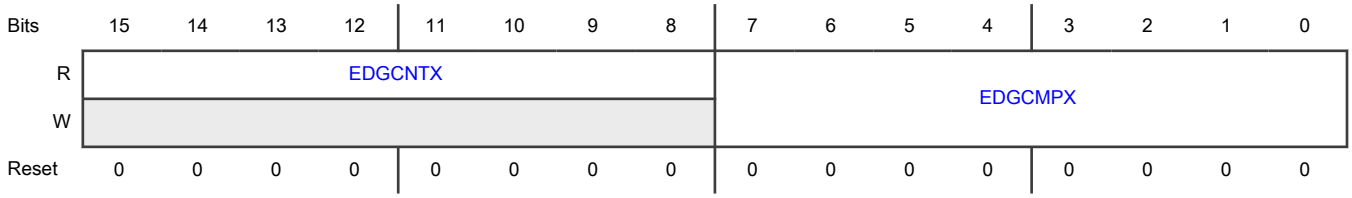
Offset

Register	Offset
SM0CAPTCOMPX	3Eh

Function

Contains capture and control values for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTX	Edge Counter X This field contains the edge counter value for the PWM_X input capture circuitry.
7-0 EDGCMPX	Edge Compare X This field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

54.5.32 Capture Value 0 Register (SM0CVAL0)

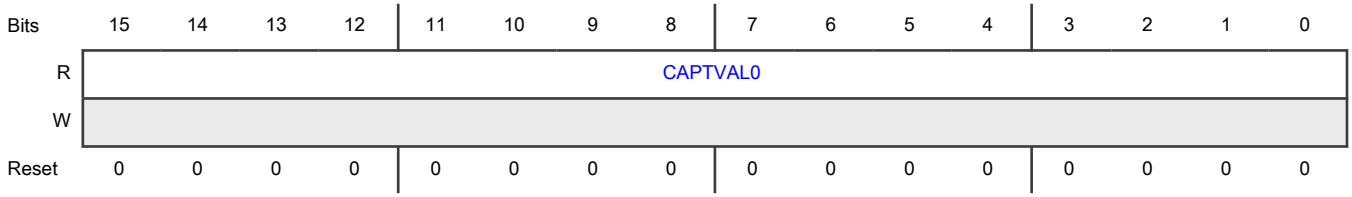
Offset

Register	Offset
SM0CVAL0	40h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL0	Capture Value 0 This field stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture increases the value of CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This field is not byte accessible.

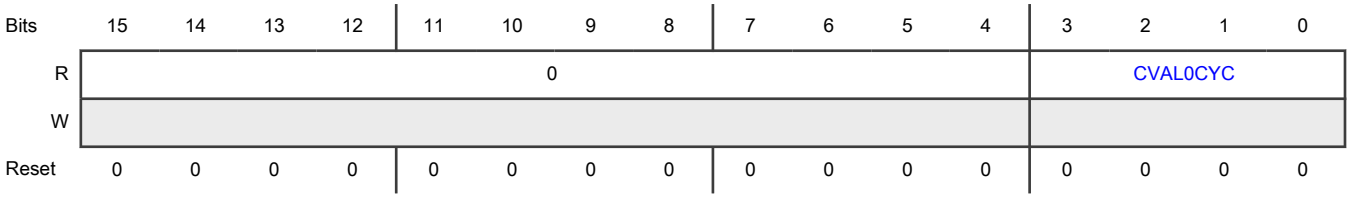
54.5.33 Capture Value 0 Cycle Register (SM0CVAL0CYC)

Offset

Register	Offset
SM0CVAL0CYC	42h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL0CYC	Capture Value 0 Cycle This field stores the cycle number corresponding to the value captured in CVAL0. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

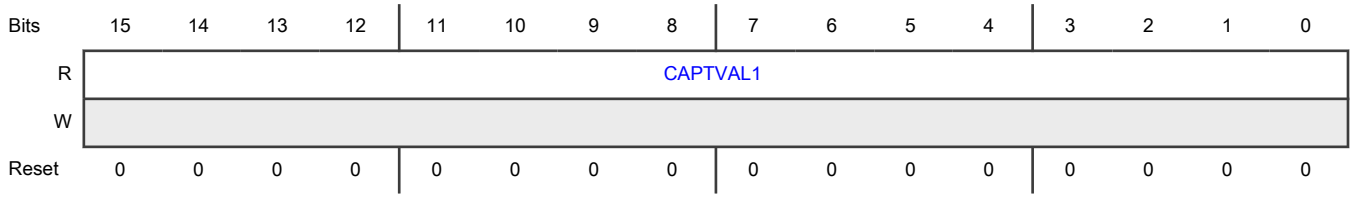
54.5.34 Capture Value 1 Register (SM0CVAL1)

Offset

Register	Offset
SM0CVAL1	44h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL1	Capture Value 1 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.35 Capture Value 1 Cycle Register (SM0CVAL1CYC)

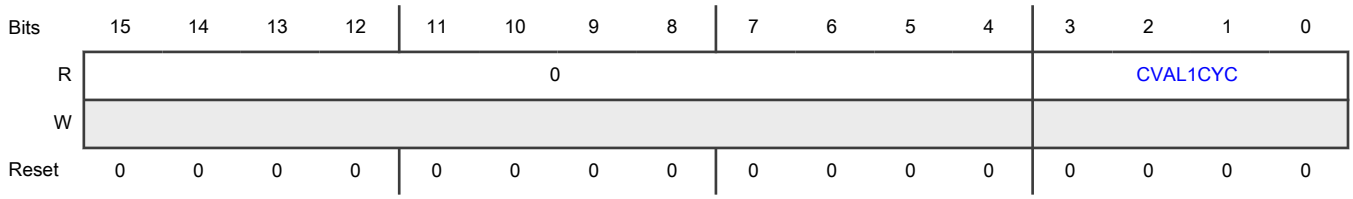
Offset

Register	Offset
SM0CVAL1CYC	46h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL1CYC	Capture Value 1 Cycle This field stores the cycle number corresponding to the value captured in CVAL1. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle. Resets to 0 at POR or hard reset.

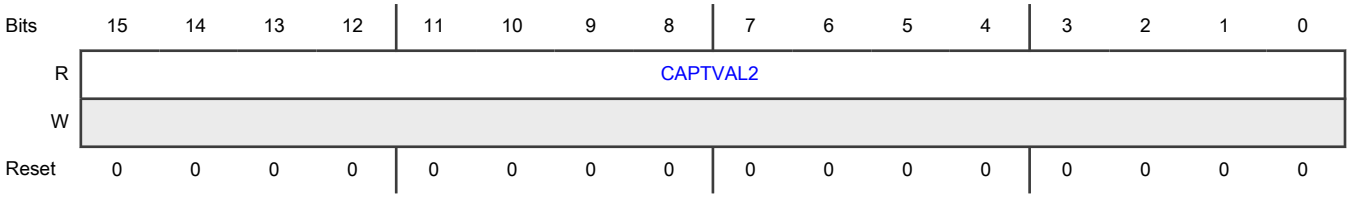
54.5.36 Capture Value 2 Register (SM0CVAL2)

Offset

Register	Offset
SM0CVAL2	48h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL2	Capture Value 2 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This field is not byte accessible.

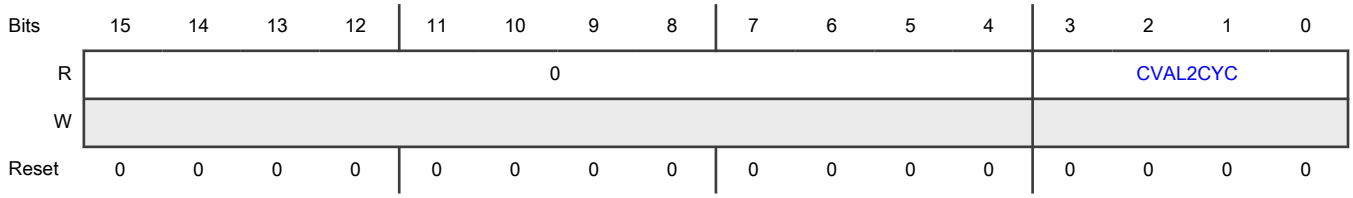
54.5.37 Capture Value 2 Cycle Register (SM0CVAL2CYC)

Offset

Register	Offset
SM0CVAL2CYC	4Ah

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL2CYC	Capture Value 2 Cycle This field stores the cycle number corresponding to the value captured in CVAL2. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.38 Capture Value 3 Register (SM0CVAL3)

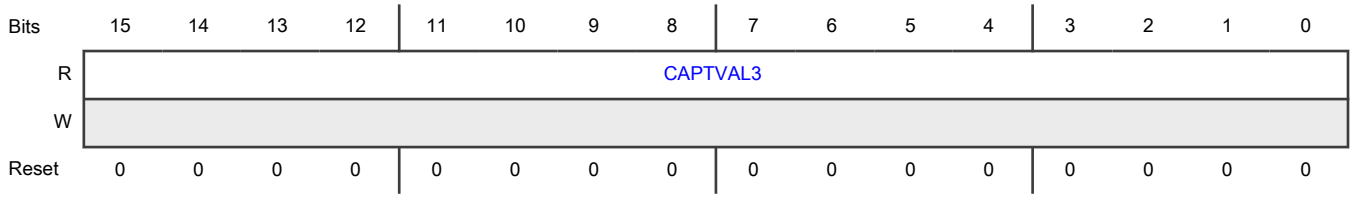
Offset

Register	Offset
SM0CVAL3	4Ch

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL3	Capture Value 3 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This field is not byte accessible.

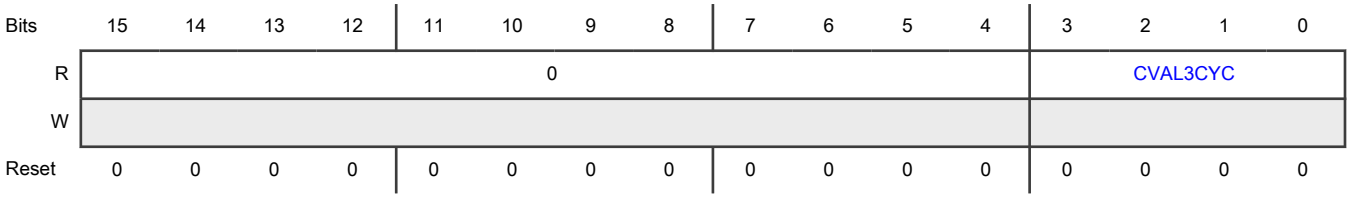
54.5.39 Capture Value 3 Cycle Register (SM0CVAL3CYC)

Offset

Register	Offset
SM0CVAL3CYC	4Eh

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL3CYC	Capture Value 3 Cycle This field stores the cycle number corresponding to the value captured in CVAL3. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

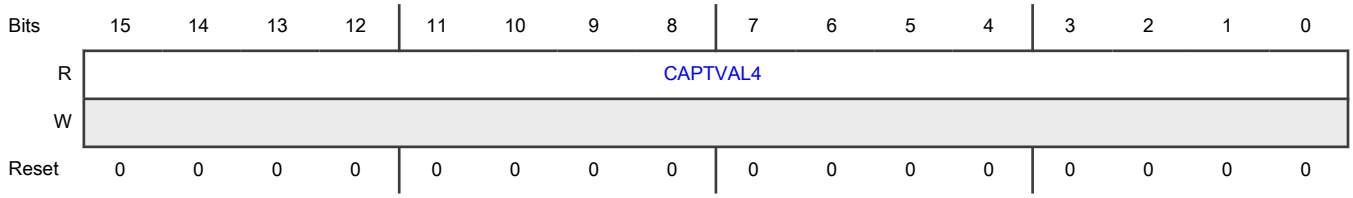
54.5.40 Capture Value 4 Register (SM0CVAL4)

Offset

Register	Offset
SM0CVAL4	50h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL4	Capture Value 4 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.41 Capture Value 4 Cycle Register (SM0CVAL4CYC)

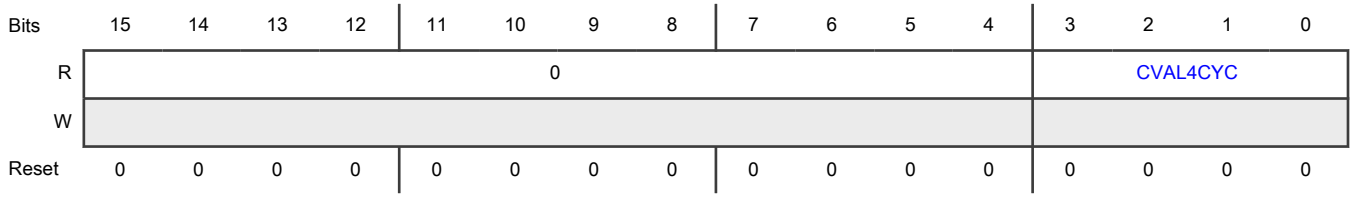
Offset

Register	Offset
SM0CVAL4CYC	52h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL4CYC	Capture Value 4 Cycle This field stores the cycle number corresponding to the value captured in CVAL4. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

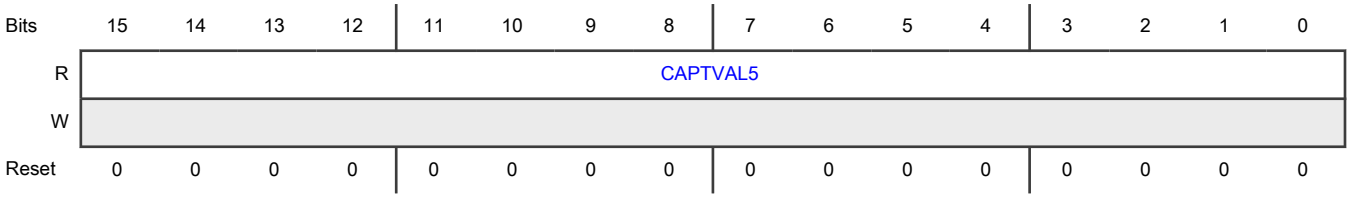
54.5.42 Capture Value 5 Register (SM0CVAL5)

Offset

Register	Offset
SM0CVAL5	54h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL5	Capture Value 5 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This field is not byte accessible.

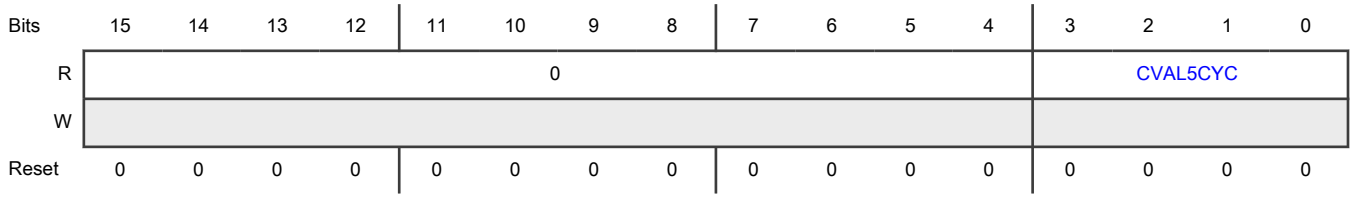
54.5.43 Capture Value 5 Cycle Register (SM0CVAL5CYC)

Offset

Register	Offset
SM0CVAL5CYC	56h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL5CYC	Capture Value 5 Cycle This field stores the cycle number corresponding to the value captured in CVAL5. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.44 Capture PWM_A Input Filter Register (SM0CAPTFILTA - SM3CAPTFILTA)

Offset

Register	Offset
SM0CAPTFILTA	5Ah
SM1CAPTFILTA	BAh
SM2CAPTFILTA	11Ah
SM3CAPTFILTA	17Ah

Function

Input filter considerations include:

- The CAPTA_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTA_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTA_FILT_CNT+3 power.
- The values of CAPTA_FILT_PER and CAPTA_FILT_CNT must be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTA_FILT_PER to a non-zero value) introduces a latency of ((CAPTA_FILT_CNT+4) x CAPTA_FILT_PER x IPBus clock period).

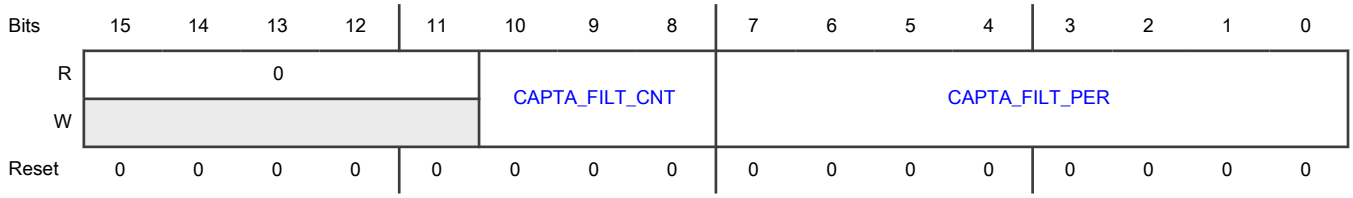
NOTE

When the filter is enabled, there is a combinational path to disable the PWM outputs to ensure rapid response to input capture conditions if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is not byte accessible

This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-8 CAPTA_FILT_CNT	Input Capture Filter Count This field represents the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTA_FILT_CNT affects the input latency.
7-0 CAPTA_FILT_PER	Input Capture Filter Period This field applies universally to all capture inputs. These bits represent the sampling period (in IPBus clock cycles) of the input capture pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTA_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTA_FILT_PER affects the input latency. **NOTE** When changing values for CAPTA_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

54.5.45 Capture PWM_B Input Filter Register (SM0CAPTFILTB - SM3CAPTFILTB)

Offset

Register	Offset
SM0CAPTFILTB	5Ch
SM1CAPTFILTB	BCh
SM2CAPTFILTB	11Ch
SM3CAPTFILTB	17Ch

Function

Input filter considerations include:

- The CAPTB_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTB_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTB_FILT_CNT+3 power.

- The values of CAPTB_FILT_PER and CAPTB_FILT_CNT must be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTB_FILT_PER to a non-zero value) introduces a latency of ((CAPTB_FILT_CNT+4) x CAPTB_FILT_PER x IPBus clock period).

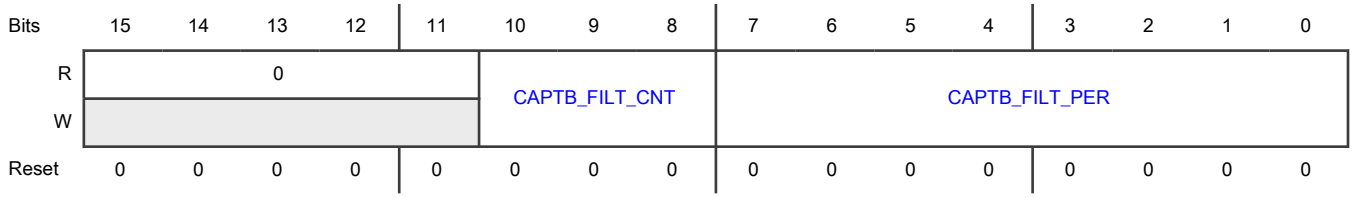
NOTE

When the filter is enabled, there is a combinational path to disable the PWM outputs to ensure rapid response to input capture conditions if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is not byte accessible

This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-8 CAPTB_FILT_CNT	Input Capture Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTB_FILT_CNT affects the input latency.
7-0 CAPTB_FILT_PER	Input Capture Filter Period This field applies universally to all capture inputs. These bits represent the sampling period (in IPBus clock cycles) of the input capture pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTB_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTB_FILT_PER affects the input latency. **NOTE** When changing values for CAPTB_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

54.5.46 Capture PWM_X Input Filter Register (SM0CAPTFILTX - SM3CAPTFILTX)

Offset

Register	Offset
SM0CAPTFILTX	5Eh
SM1CAPTFILTX	BEh
SM2CAPTFILTX	11Eh
SM3CAPTFILTX	17Eh

Function

Input filter considerations include:

- The CAPTX_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTX_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTX_FILT_CNT+3 power.
- The values of FILT_PER and CAPTX_FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTX_FILT_PER to a non-zero value) introduces a latency of ((CAPTX_FILT_CNT+4) x CAPTX_FILT_PER x IPBus clock period).

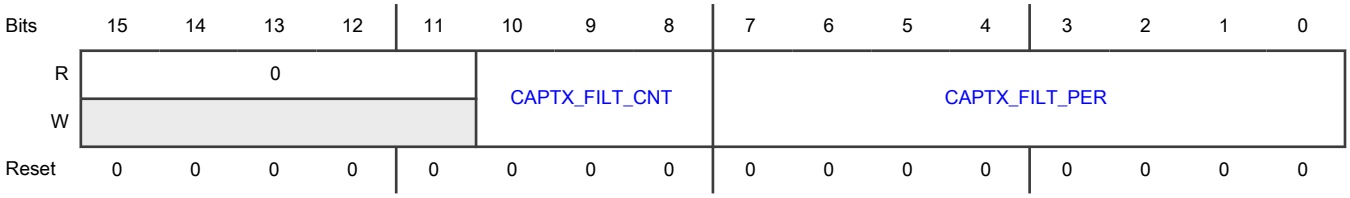
NOTE

When the filter is enabled, there is a combinational path to disable the PWM outputs to ensure rapid response to input capture conditions if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is not byte accessible

This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-8	Input Capture Filter Count

Table continues on the next page...

Table continued from the previous page...

Field	Function
CAPTX_FILT_CNT	These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTX_FILT_CNT affects the input latency.
7-0 CAPTX_FILT_PER	Input Capture Filter Period This field applies universally to all capture inputs. These bits represent the sampling period (in IPBus clock cycles) of the input capture pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTX_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTX_FILT_PER affects the input latency. **NOTE** When changing values for CAPTX_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

54.5.47 Capture Control A Register (SM1CAPTCTRLA)

Offset

Register	Offset
SM1CAPTCTRLA	94h

Function

Contains capture controls for mode A. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CA1CNT			CA0CNT			CFAWM		EDGCNTA...	INP_SELA	EDGA1		EDGA0		ONESHOTA	ARMA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1.)
12-10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1.)

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTA_EN	Edge Counter A Enable This field enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELA	Input Select A This field selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_A input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields to enable one or both of the capture registers.
5-4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is

Table continues on the next page...

Table continued from the previous page...

Field	Function
	disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLA[ARMA] is cleared. No further captures are performed until CAPTCTRLA[ARMA] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

54.5.48 Capture Compare A Register (SM1CAPTCOMPA)

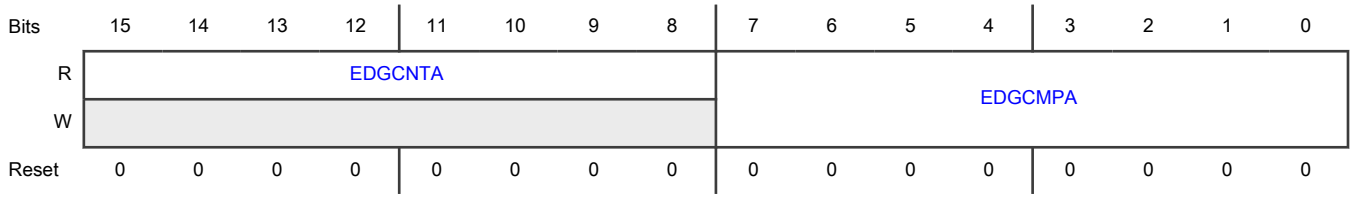
Offset

Register	Offset
SM1CAPTCOMPA	96h

Function

Contains capture and compare values for mode A.This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTA	Edge Counter A This field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

54.5.49 Capture Control B Register (SM1CAPTCTRLB)

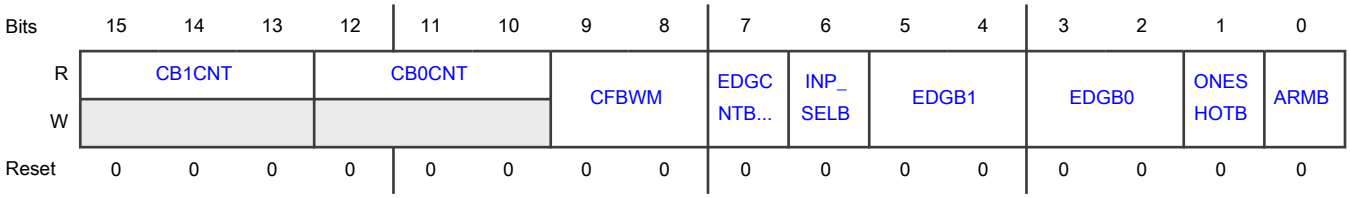
Offset

Register	Offset
SM1CAPTCTRLB	98h

Function

Contains capture controls for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1.)
12-10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1.)
9-8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTB_EN	Edge Counter B Enable This field enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELB	Input Select B This field selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_B input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the

Table continues on the next page...

Table continued from the previous page...

Field	Function
	CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields to enable one or both of the capture registers.
5-4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLB[ARMB] is cleared. No further captures are performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

54.5.50 Capture Compare B Register (SM1CAPTCOMPB)

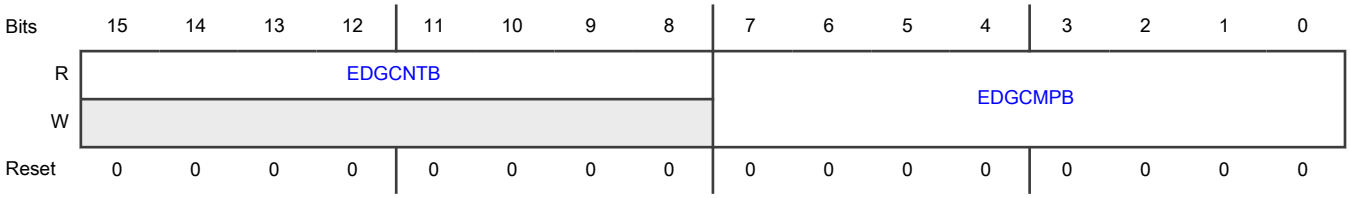
Offset

Register	Offset
SM1CAPTCOMPB	9Ah

Function

Contains capture and compare values for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTB	Edge Counter B This field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

54.5.51 Capture Control X Register (SM1CAPTCTRLX)

Offset

Register	Offset
SM1CAPTCTRLX	9Ch

Function

Contains capture controls for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CX1CNT				CX0CNT				EDGCNTX...	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
W							CFXWM									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1.)
12-10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1.)
9-8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_X input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields to enable one or both of the capture registers.
5-4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2	Edge X 0

Table continues on the next page...

Table continued from the previous page...

Field	Function
EDGX0	These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTX	One Shot Mode Aux This field selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and the ARMX bit is cleared. No further captures are performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

54.5.52 Capture Compare X Register (SM1CAPTCOMPX)

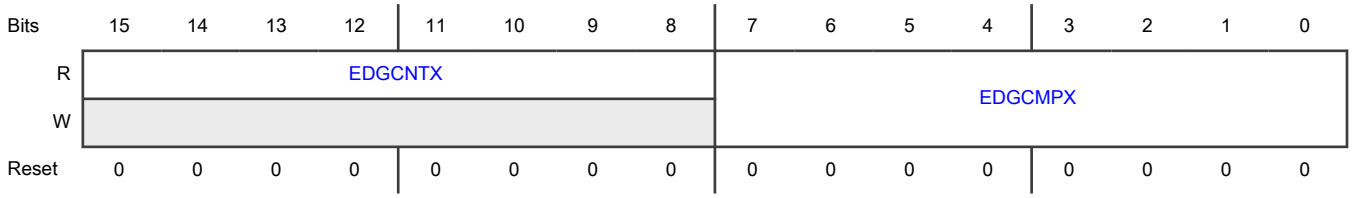
Offset

Register	Offset
SM1CAPTCOMPX	9Eh

Function

Contains capture and control values for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTX	Edge Counter X This field contains the edge counter value for the PWM_X input capture circuitry.
7-0 EDGCMPX	Edge Compare X This field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

54.5.53 Capture Value 0 Register (SM1CVAL0)

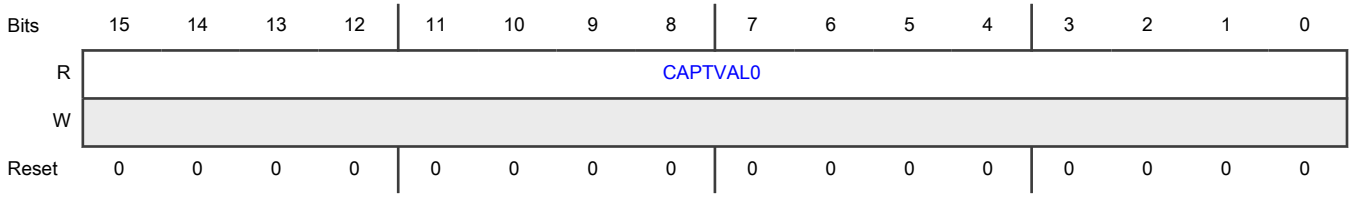
Offset

Register	Offset
SM1CVAL0	A0h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL0	Capture Value 0 This field stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture increases the value of CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This field is not byte accessible.

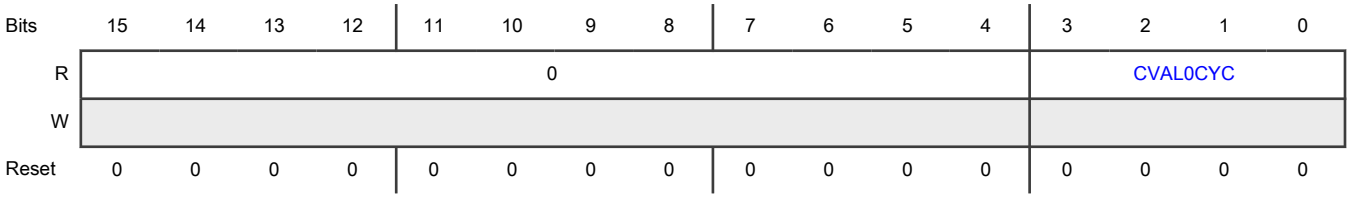
54.5.54 Capture Value 0 Cycle Register (SM1CVAL0CYC)

Offset

Register	Offset
SM1CVAL0CYC	A2h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL0CYC	Capture Value 0 Cycle This field stores the cycle number corresponding to the value captured in CVAL0. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.55 Capture Value 1 Register (SM1CVAL1)

Offset

Register	Offset
SM1CVAL1	A4h

Function
Writing this register generates bus transfer error.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-0 CAPTVAL1	Capture Value 1 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.56 Capture Value 1 Cycle Register (SM1CVAL1CYC)

Offset

Register	Offset
SM1CVAL1CYC	A6h

Function

Writing this register generates bus transfer error.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL1CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-4 —	Reserved
3-0 CVAL1CYC	Capture Value 1 Cycle This field stores the cycle number corresponding to the value captured in CVAL1. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle. Resets to 0 at POR or hard reset.

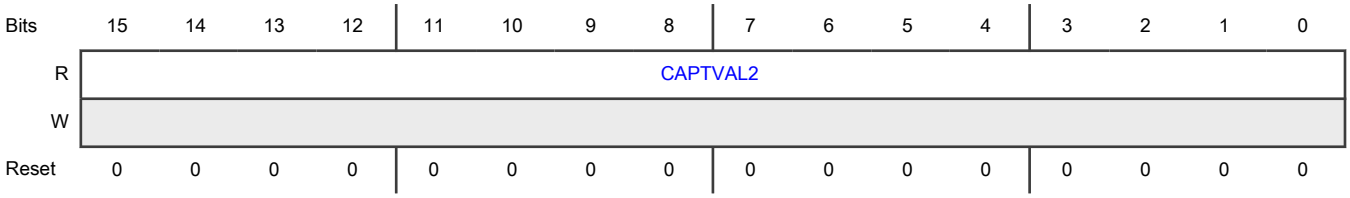
54.5.57 Capture Value 2 Register (SM1CVAL2)

Offset

Register	Offset
SM1CVAL2	A8h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL2	Capture Value 2 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This field is not byte accessible.

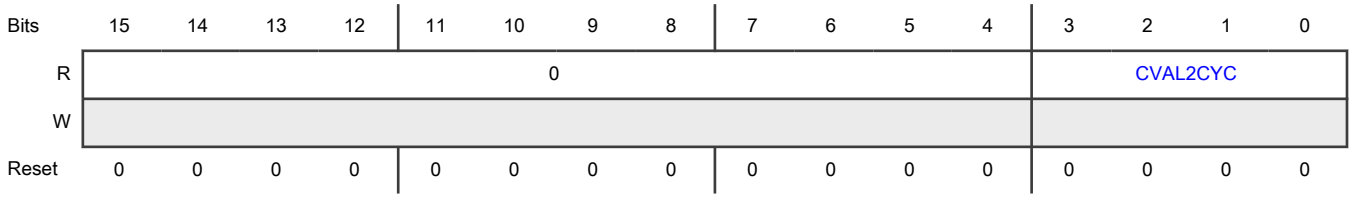
54.5.58 Capture Value 2 Cycle Register (SM1CVAL2CYC)

Offset

Register	Offset
SM1CVAL2CYC	AAh

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL2CYC	Capture Value 2 Cycle This field stores the cycle number corresponding to the value captured in CVAL2. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.59 Capture Value 3 Register (SM1CVAL3)

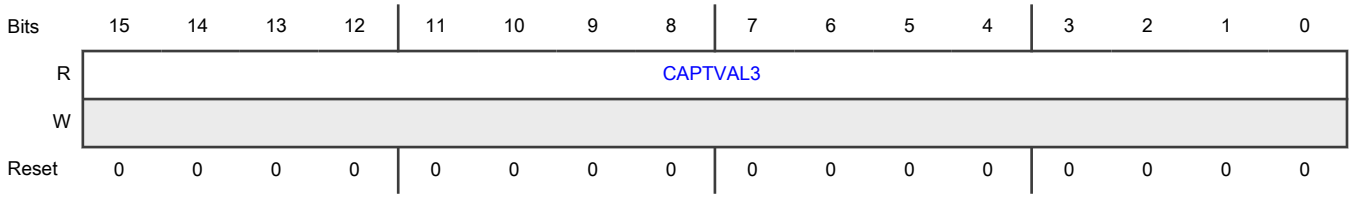
Offset

Register	Offset
SM1CVAL3	ACh

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL3	Capture Value 3 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This field is not byte accessible.

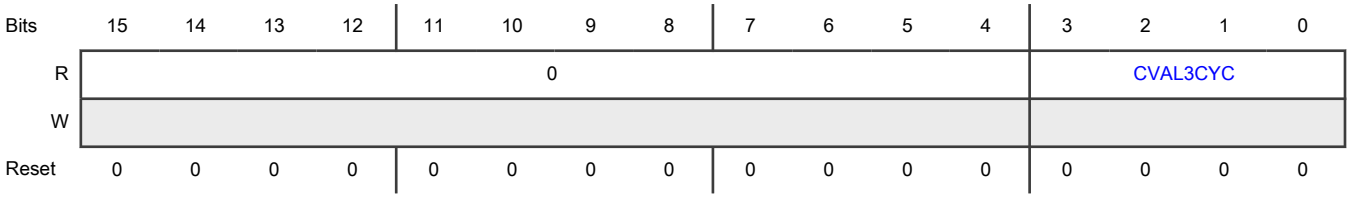
54.5.60 Capture Value 3 Cycle Register (SM1CVAL3CYC)

Offset

Register	Offset
SM1CVAL3CYC	AEh

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL3CYC	Capture Value 3 Cycle This field stores the cycle number corresponding to the value captured in CVAL3. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

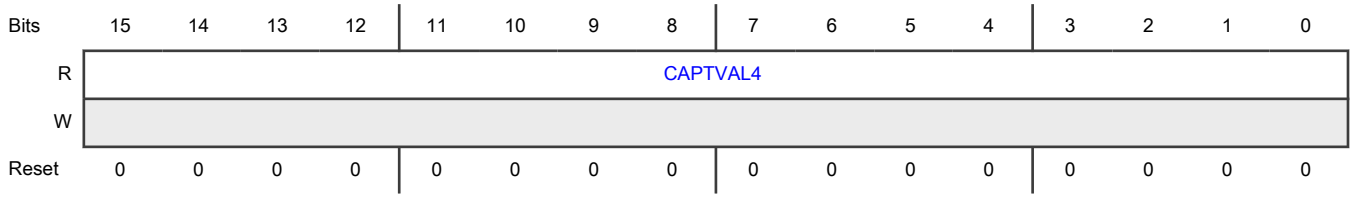
54.5.61 Capture Value 4 Register (SM1CVAL4)

Offset

Register	Offset
SM1CVAL4	B0h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL4	Capture Value 4 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.62 Capture Value 4 Cycle Register (SM1CVAL4CYC)

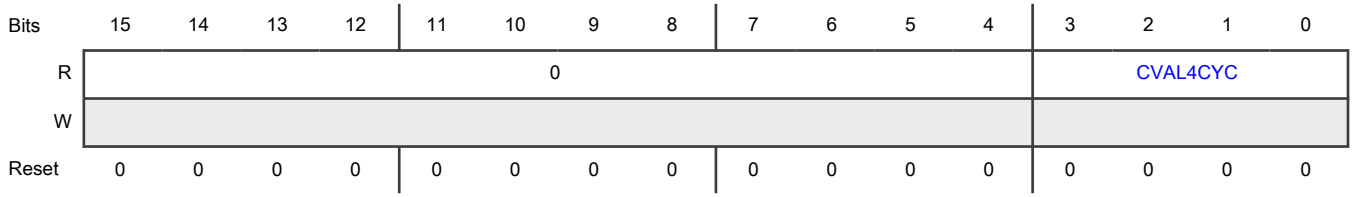
Offset

Register	Offset
SM1CVAL4CYC	B2h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL4CYC	Capture Value 4 Cycle This field stores the cycle number corresponding to the value captured in CVAL4. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

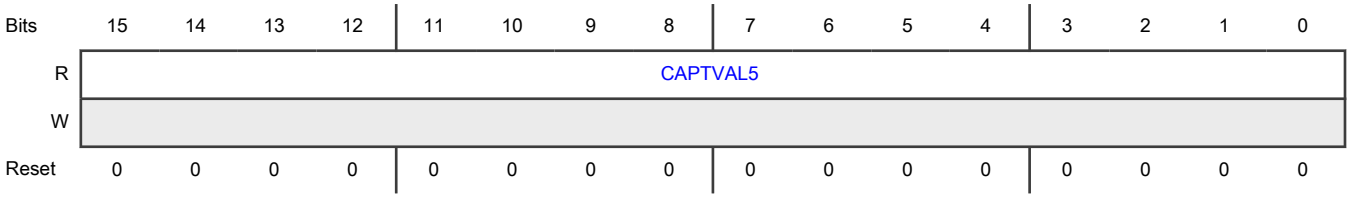
54.5.63 Capture Value 5 Register (SM1CVAL5)

Offset

Register	Offset
SM1CVAL5	B4h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL5	Capture Value 5 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This field is not byte accessible.

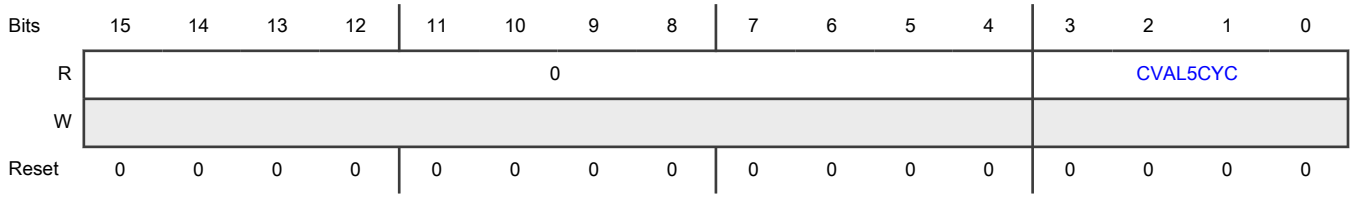
54.5.64 Capture Value 5 Cycle Register (SM1CVAL5CYC)

Offset

Register	Offset
SM1CVAL5CYC	B6h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL5CYC	Capture Value 5 Cycle This field stores the cycle number corresponding to the value captured in CVAL5. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.65 Phase Delay Register (SM1PHASEDLY - SM3PHASEDLY)

Offset

Register	Offset
SM1PHASEDLY	B8h
SM2PHASEDLY	118h
SM3PHASEDLY	178h

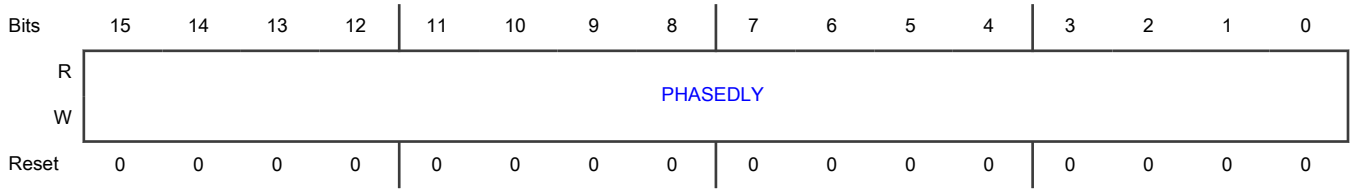
Function

The 16-bit unsigned value in this buffered register defines the delay from the master sync signal of submodule 0 to the time that this submodule recognizes the master sync in PWM clock periods. CTRL2[INIT_SEL] must be set to 10b to select the master sync signal as the source for initialization when using this register. Setting this register with a non-zero value and using the master sync signal as the initialization source, allows the output of this submodule to be a fixed number of cycles delayed from submodule 0. For PWM operation, the buffered contents of this register are updated at the start of every PWM cycle. This field is not byte accessible.

NOTE

The PHASEDLY register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading PHASEDLY reads the value in a buffer and not necessarily the value that the PWM generator is currently using. Also, the value of this register should not be set to a value larger than the period defined in submodule 0.

Diagram



Fields

Field	Function
15-0 PHASEDLY	Initial Count Register Bits

54.5.66 Capture Control A Register (SM2CAPTCTRLA)

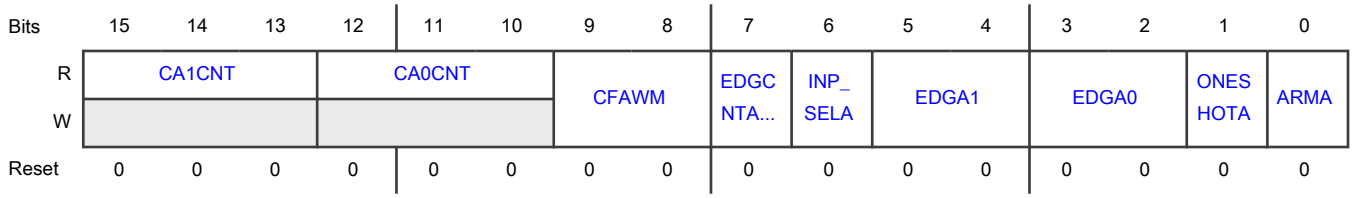
Offset

Register	Offset
SM2CAPTCTRLA	F4h

Function

Contains capture controls for mode A. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1.)
12-10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1.)
9-8 CFAWM	Capture A FIFOs Water Mark

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTA_EN	Edge Counter A Enable This field enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELA	Input Select A This field selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_A input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields to enable one or both of the capture registers.
5-4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLA[ARMA] is cleared. No further captures are performed until CAPTCTRLA[ARMA] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

54.5.67 Capture Compare A Register (SM2CAPTCOMPA)

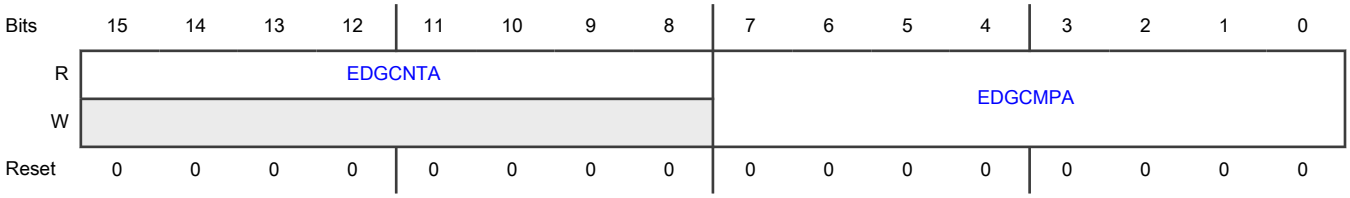
Offset

Register	Offset
SM2CAPTCOMPA	F6h

Function

Contains capture and compare values for mode A.This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTA	Edge Counter A This field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

54.5.68 Capture Control B Register (SM2CAPTCTRLB)

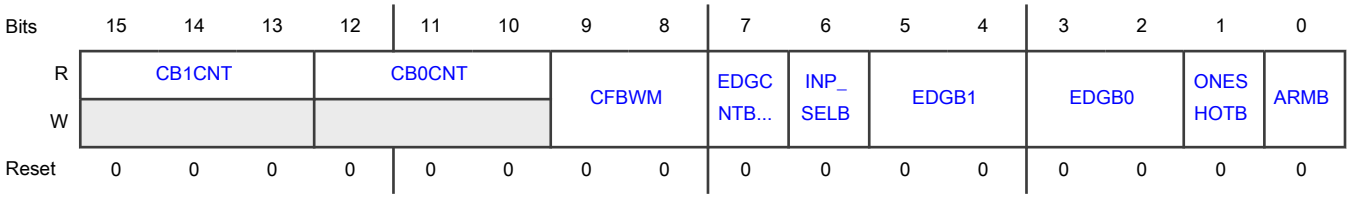
Offset

Register	Offset
SM2CAPTCTRLB	F8h

Function

Contains capture controls for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1.)
12-10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1.)
9-8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTB_EN	Edge Counter B Enable This field enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELB	Input Select B This field selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_B input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the

Table continues on the next page...

Table continued from the previous page...

Field	Function
	CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields to enable one or both of the capture registers.
5-4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLB[ARMB] is cleared. No further captures are performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

54.5.69 Capture Compare B Register (SM2CAPTCOMPB)

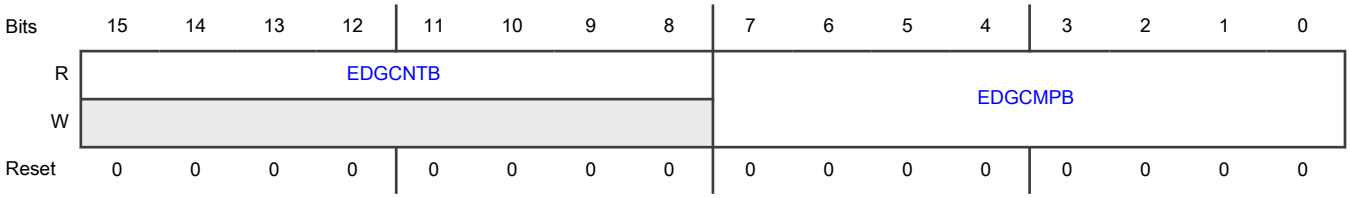
Offset

Register	Offset
SM2CAPTCOMPB	FAh

Function

Contains capture and compare values for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTB	Edge Counter B This field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

54.5.70 Capture Control X Register (SM2CAPTCTRLX)

Offset

Register	Offset
SM2CAPTCTRLX	FCh

Function

Contains capture controls for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CX1CNT				CX0CNT				EDGCNTX...	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
W							CFXWM									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1.)
12-10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1.)
9-8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_X input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields to enable one or both of the capture registers.
5-4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2	Edge X 0

Table continues on the next page...

Table continued from the previous page...

Field	Function
EDGX0	These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTX	One Shot Mode Aux This field selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and the ARMX bit is cleared. No further captures are performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

54.5.71 Capture Compare X Register (SM2CAPTCOMPX)

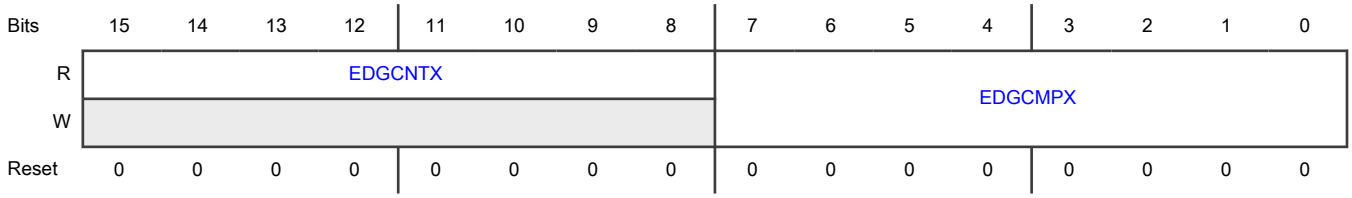
Offset

Register	Offset
SM2CAPTCOMPX	FEh

Function

Contains capture and control values for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTX	Edge Counter X This field contains the edge counter value for the PWM_X input capture circuitry.
7-0 EDGCMPX	Edge Compare X This field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

54.5.72 Capture Value 0 Register (SM2CVAL0)

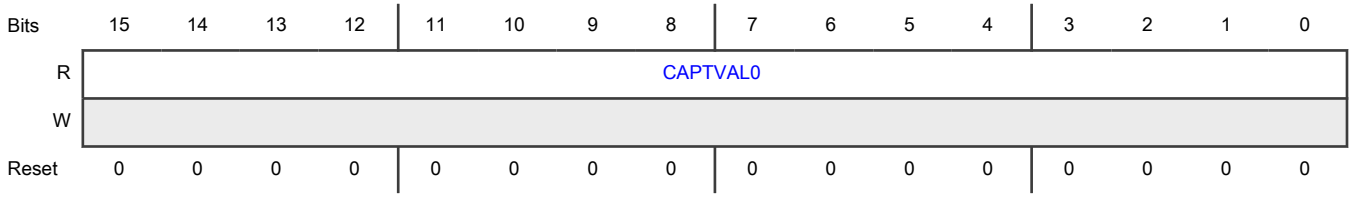
Offset

Register	Offset
SM2CVAL0	100h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL0	Capture Value 0 This field stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture increases the value of CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This field is not byte accessible.

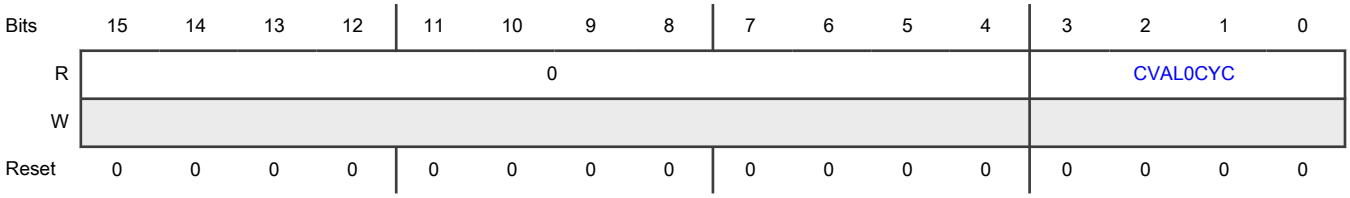
54.5.73 Capture Value 0 Cycle Register (SM2CVAL0CYC)

Offset

Register	Offset
SM2CVAL0CYC	102h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL0CYC	Capture Value 0 Cycle This field stores the cycle number corresponding to the value captured in CVAL0. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.74 Capture Value 1 Register (SM2CVAL1)

Offset

Register	Offset
SM2CVAL1	104h

Function
Writing this register generates bus transfer error.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-0 CAPTVAL1	Capture Value 1 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.75 Capture Value 1 Cycle Register (SM2CVAL1CYC)

Offset

Register	Offset
SM2CVAL1CYC	106h

Function

Writing this register generates bus transfer error.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL1CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-4 —	Reserved
3-0 CVAL1CYC	Capture Value 1 Cycle This field stores the cycle number corresponding to the value captured in CVAL1. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle. Resets to 0 at POR or hard reset.

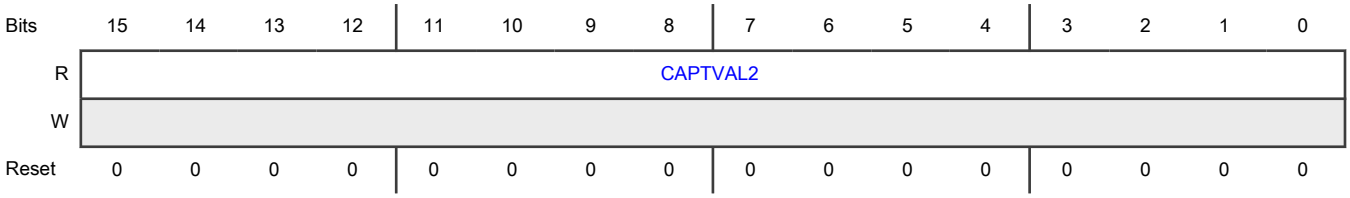
54.5.76 Capture Value 2 Register (SM2CVAL2)

Offset

Register	Offset
SM2CVAL2	108h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL2	Capture Value 2 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This field is not byte accessible.

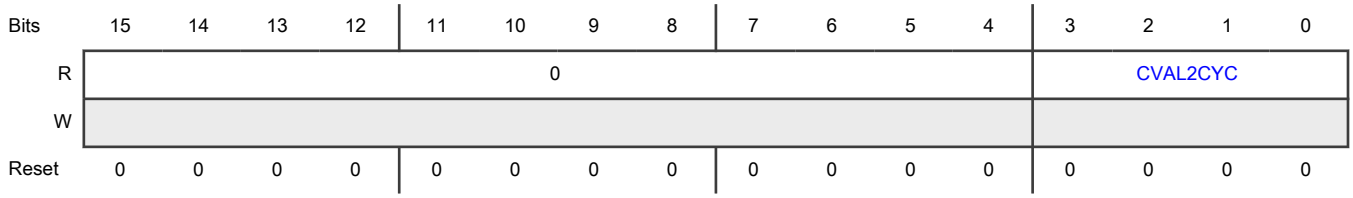
54.5.77 Capture Value 2 Cycle Register (SM2CVAL2CYC)

Offset

Register	Offset
SM2CVAL2CYC	10Ah

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL2CYC	Capture Value 2 Cycle This field stores the cycle number corresponding to the value captured in CVAL2. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.78 Capture Value 3 Register (SM2CVAL3)

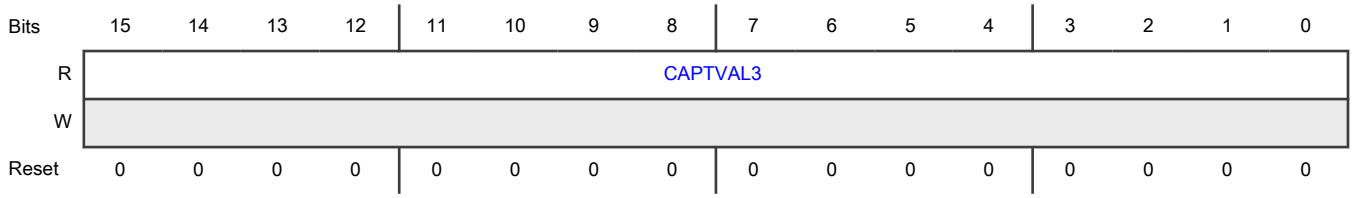
Offset

Register	Offset
SM2CVAL3	10Ch

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL3	Capture Value 3 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This field is not byte accessible.

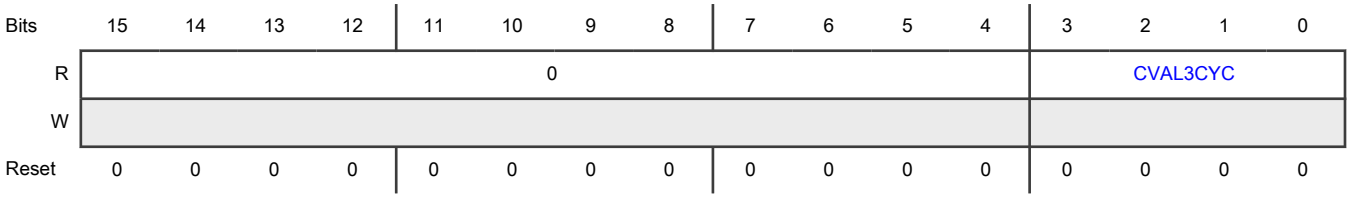
54.5.79 Capture Value 3 Cycle Register (SM2CVAL3CYC)

Offset

Register	Offset
SM2CVAL3CYC	10Eh

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL3CYC	Capture Value 3 Cycle This field stores the cycle number corresponding to the value captured in CVAL3. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

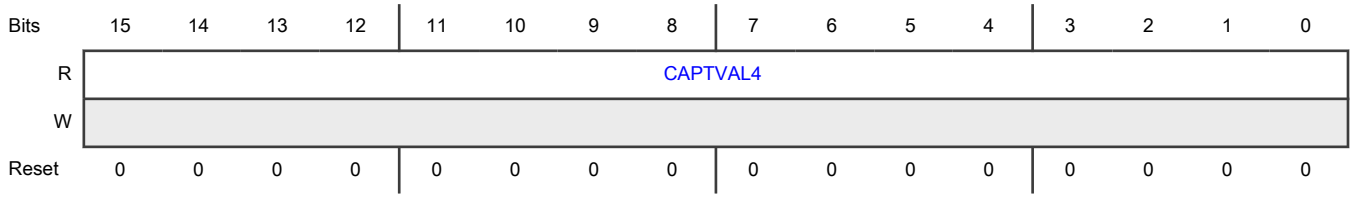
54.5.80 Capture Value 4 Register (SM2CVAL4)

Offset

Register	Offset
SM2CVAL4	110h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL4	Capture Value 4 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.81 Capture Value 4 Cycle Register (SM2CVAL4CYC)

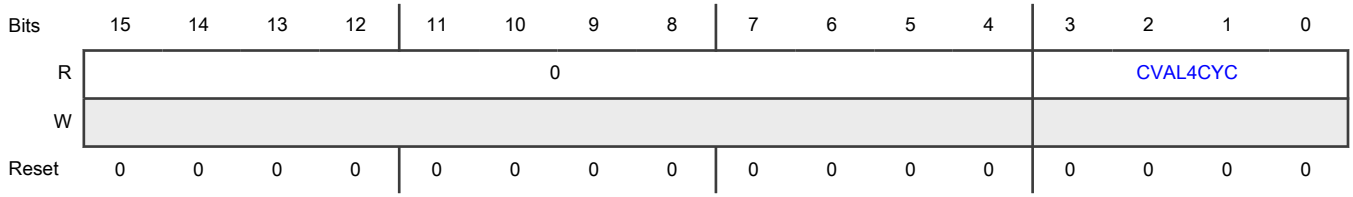
Offset

Register	Offset
SM2CVAL4CYC	112h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL4CYC	Capture Value 4 Cycle This field stores the cycle number corresponding to the value captured in CVAL4. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

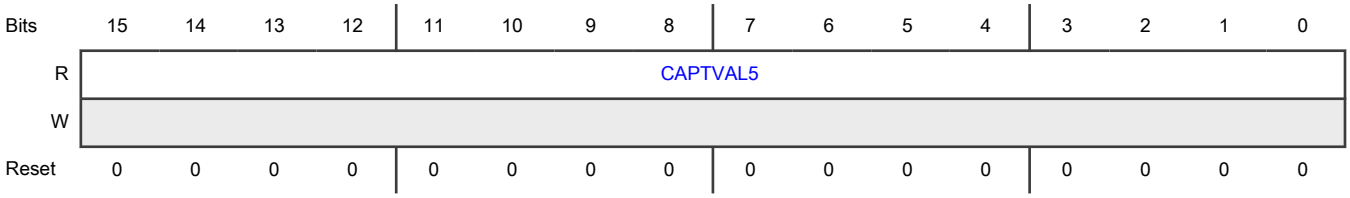
54.5.82 Capture Value 5 Register (SM2CVAL5)

Offset

Register	Offset
SM2CVAL5	114h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL5	Capture Value 5 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This field is not byte accessible.

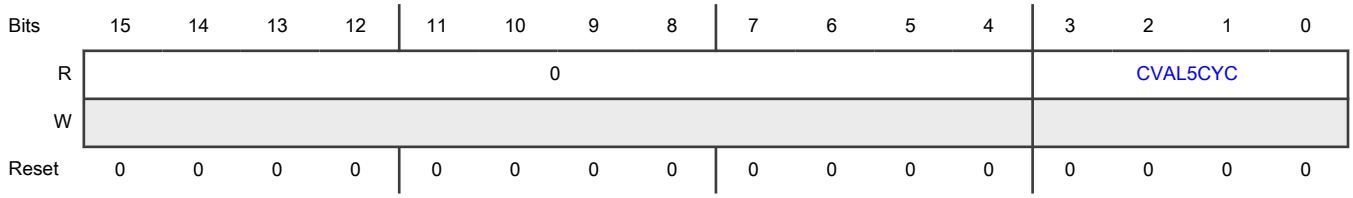
54.5.83 Capture Value 5 Cycle Register (SM2CVAL5CYC)

Offset

Register	Offset
SM2CVAL5CYC	116h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL5CYC	Capture Value 5 Cycle This field stores the cycle number corresponding to the value captured in CVAL5. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.84 Capture Control A Register (SM3CAPTCTRLA)

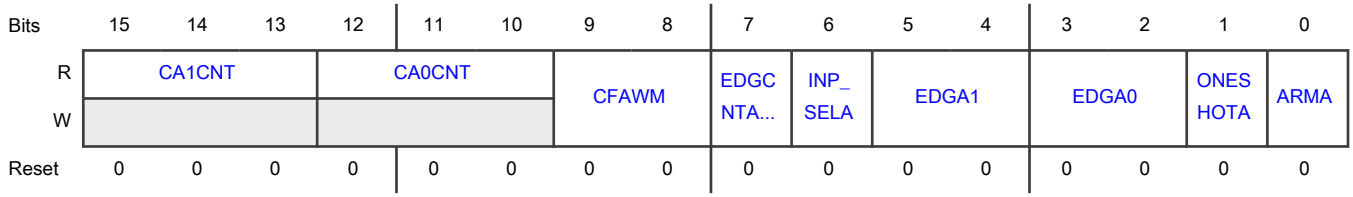
Offset

Register	Offset
SM3CAPTCTRLA	154h

Function

Contains capture controls for mode A. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1.)
12-10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1.)

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTA_EN	Edge Counter A Enable This field enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELA	Input Select A This field selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_A input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields to enable one or both of the capture registers.
5-4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is

Table continues on the next page...

Table continued from the previous page...

Field	Function
	disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLA[ARMA] is cleared. No further captures are performed until CAPTCTRLA[ARMA] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

54.5.85 Capture Compare A Register (SM3CAPTCOMPA)

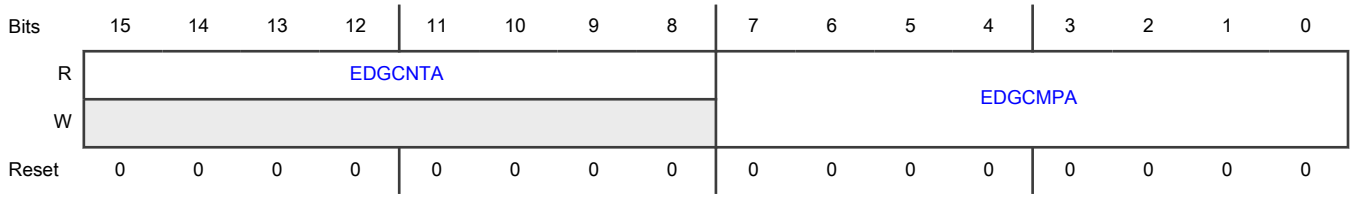
Offset

Register	Offset
SM3CAPTCOMPA	156h

Function

Contains capture and compare values for mode A.This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTA	Edge Counter A This field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

54.5.86 Capture Control B Register (SM3CAPTCTRLB)

Offset

Register	Offset
SM3CAPTCTRLB	158h

Function

Contains capture controls for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CB1CNT			CB0CNT			CFBWM		EDGCNTB...	INP_SELB	EDGB1		EDGB0		ONESHOTB	ARMB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1.)
12-10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1.)
9-8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTB_EN	Edge Counter B Enable This field enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELB	Input Select B This field selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_B input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the

Table continues on the next page...

Table continued from the previous page...

Field	Function
	CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields to enable one or both of the capture registers.
5-4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and CAPTCTRLB[ARMB] is cleared. No further captures are performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

54.5.87 Capture Compare B Register (SM3CAPTCOMPB)

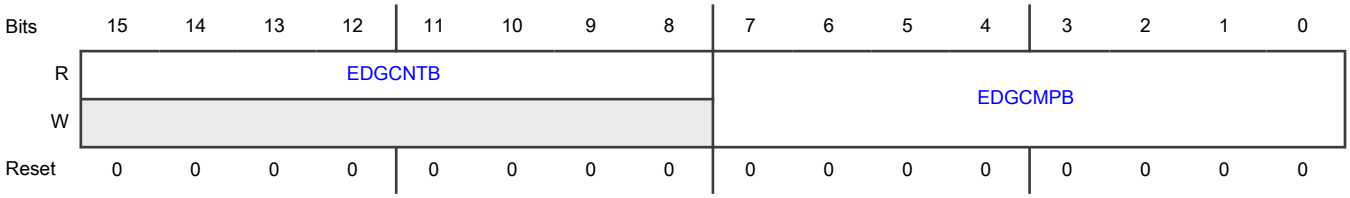
Offset

Register	Offset
SM3CAPTCOMPB	15Ah

Function

Contains capture and compare values for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTB	Edge Counter B This field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

54.5.88 Capture Control X Register (SM3CAPTCTRLX)

Offset

Register	Offset
SM3CAPTCTRLX	15Ch

Function

Contains capture controls for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CX1CNT			CX0CNT			CFXWM		EDGCNTX...	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1.)
12-10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1.)
9-8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], will not be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_X input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. When this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields to enable one or both of the capture registers.
5-4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2	Edge X 0

Table continues on the next page...

Table continued from the previous page...

Field	Function
EDGX0	These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTX	One Shot Mode Aux This field selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected if both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed, and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed, and the ARMX bit is cleared. No further captures are performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture occurs on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

54.5.89 Capture Compare X Register (SM3CAPTCOMPX)

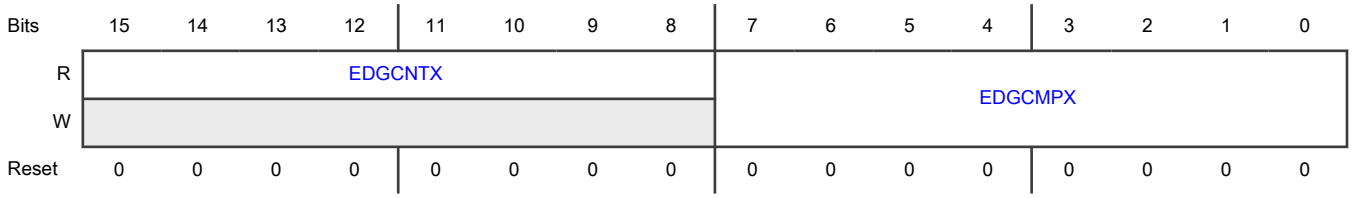
Offset

Register	Offset
SM3CAPTCOMPX	15Eh

Function

Contains capture and control values for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-8 EDGCNTX	Edge Counter X This field contains the edge counter value for the PWM_X input capture circuitry.
7-0 EDGCMPX	Edge Compare X This field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

54.5.90 Capture Value 0 Register (SM3CVAL0)

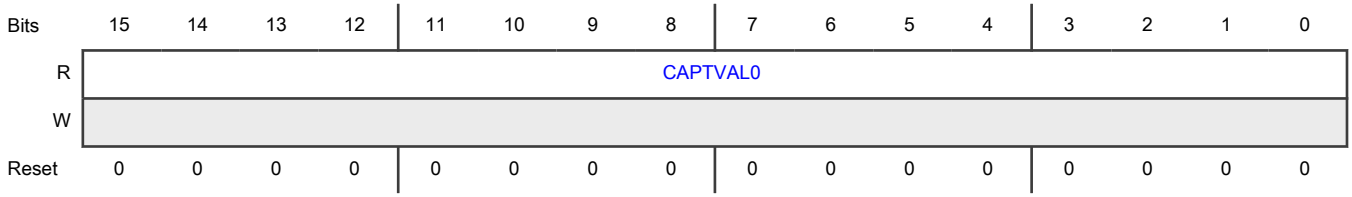
Offset

Register	Offset
SM3CVAL0	160h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL0	Capture Value 0 This field stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture increases the value of CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This field is not byte accessible.

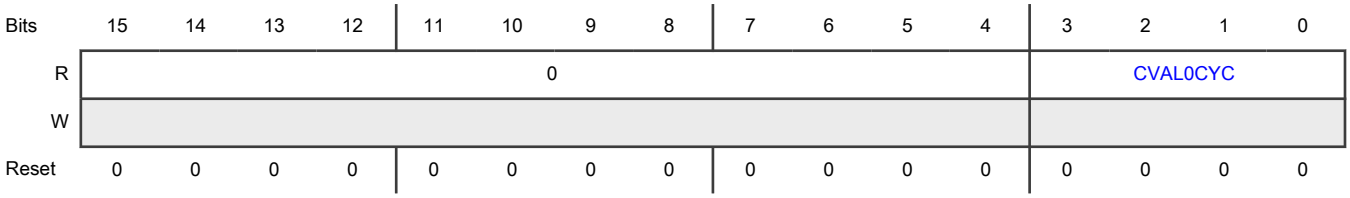
54.5.91 Capture Value 0 Cycle Register (SM3CVAL0CYC)

Offset

Register	Offset
SM3CVAL0CYC	162h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL0CYC	Capture Value 0 Cycle This field stores the cycle number corresponding to the value captured in CVAL0. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

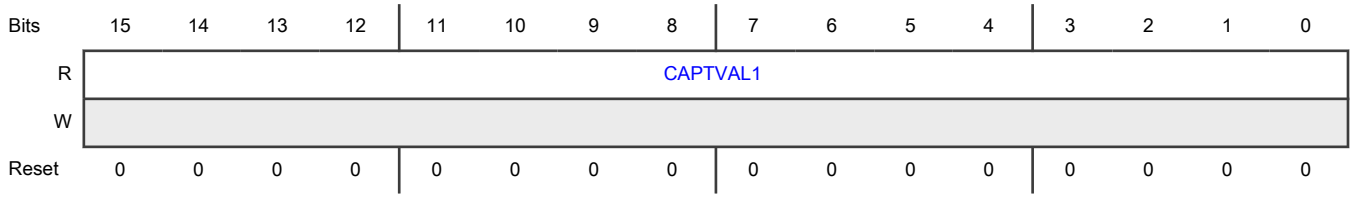
54.5.92 Capture Value 1 Register (SM3CVAL1)

Offset

Register	Offset
SM3CVAL1	164h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL1	Capture Value 1 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.93 Capture Value 1 Cycle Register (SM3CVAL1CYC)

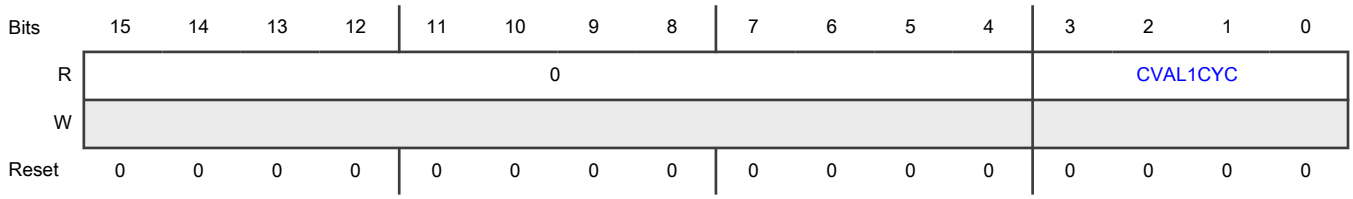
Offset

Register	Offset
SM3CVAL1CYC	166h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL1CYC	Capture Value 1 Cycle This field stores the cycle number corresponding to the value captured in CVAL1. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle. Resets to 0 at POR or hard reset.

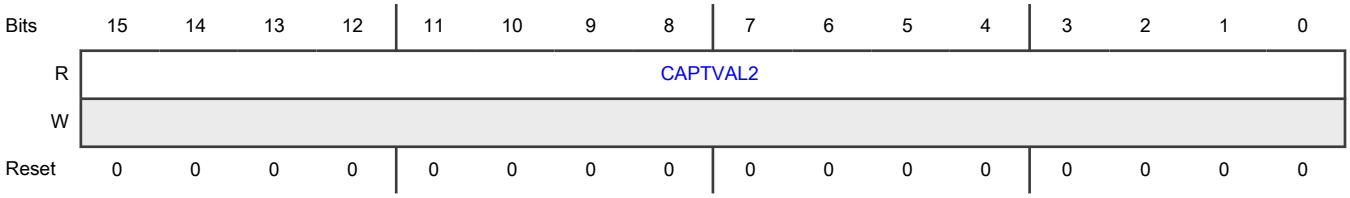
54.5.94 Capture Value 2 Register (SM3CVAL2)

Offset

Register	Offset
SM3CVAL2	168h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL2	Capture Value 2 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This field is not byte accessible.

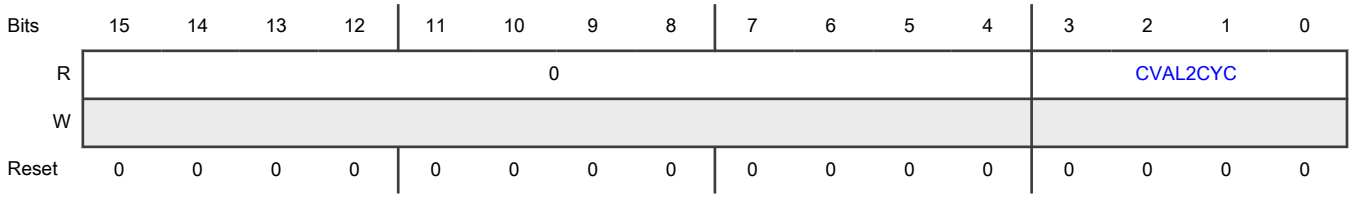
54.5.95 Capture Value 2 Cycle Register (SM3CVAL2CYC)

Offset

Register	Offset
SM3CVAL2CYC	16Ah

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL2CYC	Capture Value 2 Cycle This field stores the cycle number corresponding to the value captured in CVAL2. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.96 Capture Value 3 Register (SM3CVAL3)

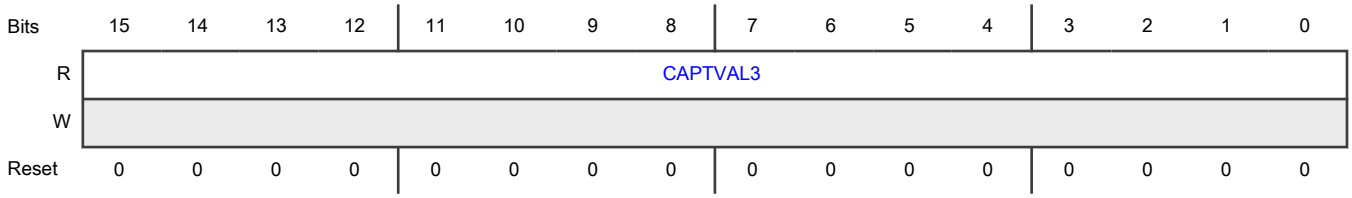
Offset

Register	Offset
SM3CVAL3	16Ch

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL3	Capture Value 3 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This field is not byte accessible.

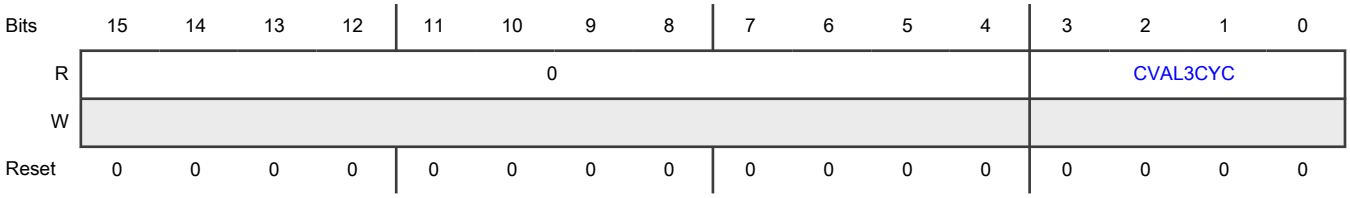
54.5.97 Capture Value 3 Cycle Register (SM3CVAL3CYC)

Offset

Register	Offset
SM3CVAL3CYC	16Eh

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL3CYC	Capture Value 3 Cycle This field stores the cycle number corresponding to the value captured in CVAL3. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

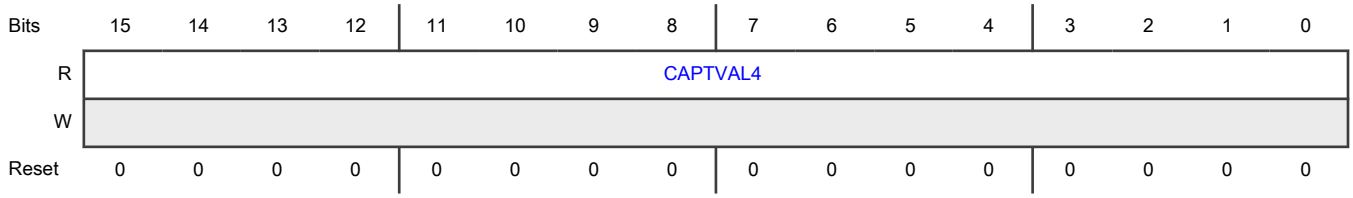
54.5.98 Capture Value 4 Register (SM3CVAL4)

Offset

Register	Offset
SM3CVAL4	170h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL4	Capture Value 4 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.99 Capture Value 4 Cycle Register (SM3CVAL4CYC)

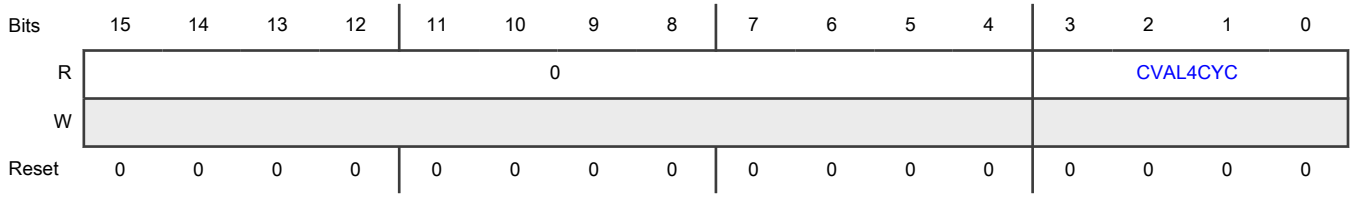
Offset

Register	Offset
SM3CVAL4CYC	172h

Function

Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-4 —	Reserved
3-0 CVAL4CYC	Capture Value 4 Cycle This field stores the cycle number corresponding to the value captured in CVAL4. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

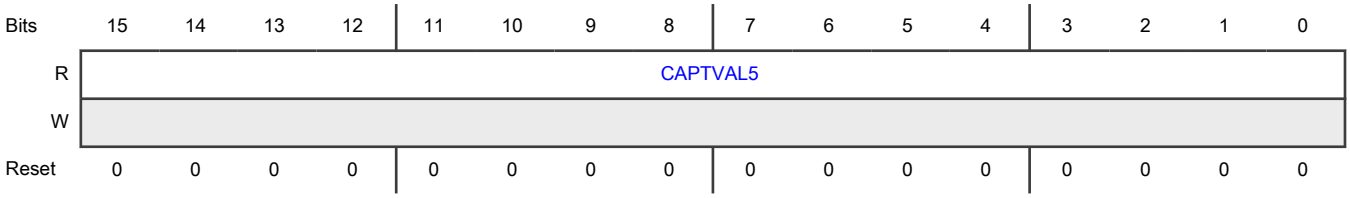
54.5.100 Capture Value 5 Register (SM3CVAL5)

Offset

Register	Offset
SM3CVAL5	174h

Function
Writing this register generates bus transfer error.

Diagram



Fields

Field	Function
15-0 CAPTVAL5	Capture Value 5 This field stores the value captured from the submodule counter when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this field decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This field is not byte accessible.

54.5.101 Capture Value 5 Cycle Register (SM3CVAL5CYC)

Offset

Register	Offset
SM3CVAL5CYC	176h

Function
Writing this register generates bus transfer error.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL5CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-4 —	Reserved
3-0 CVAL5CYC	Capture Value 5 Cycle This field stores the cycle number corresponding to the value captured in CVAL5. This field is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

54.5.102 Output Enable Register (OUTEN)

Offset

Register	Offset
OUTEN	180h

Function

Contains PWM output enables. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				PWMA_EN				PWMB_EN				PWMX_EN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-12 —	Reserved
11-8 PWMA_EN	PWM_A Output Enables

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field enables the PWM_A outputs of submodules 3 - 0, respectively. Set these bits to 0 (output disabled) when a PWM_A pin is being used for input capture. • 0b0 PWM_A output disabled. • 0b1 PWM_A output enabled.
7-4 PWMB_EN	PWM_B Output Enables This field enables the PWM_B outputs of submodules 3 - 0, respectively. Set these bits to 0 (output disabled) when a PWM_B pin is being used for input capture. • 0b0 PWM_B output disabled. • 0b1 PWM_B output enabled.
3-0 PWMX_EN	PWM_X Output Enables This field enables the PWM_X outputs of submodules 3 - 0, respectively. Set these bits to 0 (output disabled) when a PWM_X pin is being used for input capture or deadtime correction. • 0b0 PWM_X output disabled. • 0b1 PWM_X output enabled.

54.5.103 Mask Register (MASK)

Offset

Register	Offset
MASK	182h

Function

MASK is double buffered and does not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading MASK reads the buffered values and not necessarily the values currently in effect. This double buffering can be overridden by setting the UPDATE_MASK bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					MASKA				MASKB				MASKX			
W	UPDATE_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-12 UPDATE_MASK	Update Mask Bits Immediately This field masks or unmasks the PWM_X (X = A, B, or X) outputs of submodules 3 - 0 respectively by forcing the MASK* (* = A, B or X) bits to be immediately updated within submodules 3 - 0, respectively without waiting for a FORCE_OUT event. These self-clearing bits always read as zero. Software may write to any or all of these bits and may set these bits in the same write operation that updates the MASKA, MASKB, and MASKX fields of this register. • 0b0 Normal operation. MASK* bits within the corresponding submodule are not updated until a FORCE_OUT event occurs within the submodule. • 0b1 Immediate operation. MASK* bits within the corresponding submodule are updated on the following clock edge after setting this bit. For example: • UPDATE_MASK[3]=1 updates the mask bits MASKA[3], MASKB[3], and MASKX[3]. • UPDATE_MASK[3:0]=0101 updates the mask bits MASKA[2], MASKA[0], MASKB[2], MASKB[0], MASKX[2], and MASKX[0].
11-8 MASKA	PWM_A Masks This field masks the PWM_A outputs of submodules 3 - 0, respectively by forcing the output to logic 0 prior to consideration of the output polarity. • 0b0 PWM_A output normal. • 0b1 PWM_A output masked. MASKA[3:0] masks PWM_A_3 through PWM_A_0
7-4 MASKB	PWM_B Masks This field masks the PWM_B outputs of submodules 3 - 0, respectively by forcing the output to logic 0 prior to consideration of the output polarity. • 0b0 PWM_B output normal. • 0b1 PWM_B output masked. MASKB[3:0] masks PWM_B_3 through PWM_B_0
3-0 MASKX	PWM_X Masks This field masks the PWM_X outputs of submodules 3 - 0, respectively by forcing the output to logic 0 prior to consideration of the output polarity.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 0b0 PWM_X output normal. • 0b1 PWM_X output masked. MASKX[3:0] masks PWM_X_3 through PWM_X_0

54.5.104 Software Controlled Output Register (SWCOUT)

Offset

Register	Offset
SWCOUT	184h

Function

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SM3OUT23	SM3OUT45	SM2OUT23	SM2OUT45	SM1OUT23	SM1OUT45	SM0OUT23	SM0OUT45
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-8 —	Reserved
7 SM3OUT23	Submodule 3 Software Controlled Output 23 This field is used when DTSRCSEL[SM3SEL23] is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
6 SM3OUT45	Submodule 3 Software Controlled Output 45 This field is used when DTSRCSEL[SM3SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
5 SM2OUT23	Submodule 2 Software Controlled Output 23 This field is used when DTSRCSEL[SM2SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.
4 SM2OUT45	Submodule 2 Software Controlled Output 45 This field is used when DTSRCSEL[SM2SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.
3 SM1OUT23	Submodule 1 Software Controlled Output 23 This field is used when DTSRCSEL[SM1SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
2 SM1OUT45	Submodule 1 Software Controlled Output 45 This field is used when DTSRCSEL[SM1SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
1 SM0OUT23	Submodule 0 Software Controlled Output 23 This field is used when DTSRCSEL[SM0SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
0 SM0OUT45	Submodule 0 Software Controlled Output 45 This field is used when DTSRCSEL[SM0SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

54.5.105 PWM Source Select Register (DTSRCSEL)

Offset

Register	Offset
DTSRCSEL	186h

Function

The PWM source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SM3SEL23		SM3SEL45		SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23		SM0SEL45	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 SM3SEL23	Submodule 3 PWM23 Control Select This field selects possible overrides to the generated SM3PWM23 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM3PWM23 signal used by the deadtime logic. 01b - Inverted generated SM3PWM23 signal used by the deadtime logic. 10b - SWCOUT[SM3OUT23] used by the deadtime logic. 11b - PWM3_EXTA signal used by the deadtime logic.
13-12 SM3SEL45	Submodule 3 PWM45 Control Select This field selects possible overrides to the generated SM3PWM45 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM3PWM45 signal used by the deadtime logic. 01b - Inverted generated SM3PWM45 signal used by the deadtime logic. 10b - SWCOUT[SM3OUT45] used by the deadtime logic. 11b - Reserved
11-10 SM2SEL23	Submodule 2 PWM23 Control Select This field selects possible overrides to the generated SM2PWM23 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Generated SM2PWM23 signal used by the deadtime logic. 01b - Inverted generated SM2PWM23 signal used by the deadtime logic. 10b - SWCOUT[SM2OUT23] used by the deadtime logic. 11b - PWM2_EXTA signal used by the deadtime logic.
9-8 SM2SEL45	Submodule 2 PWM45 Control Select This field selects possible overrides to the generated SM2PWM45 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM2PWM45 signal used by the deadtime logic. 01b - Inverted generated SM2PWM45 signal used by the deadtime logic. 10b - SWCOUT[SM2OUT45] used by the deadtime logic. 11b - Reserved
7-6 SM1SEL23	Submodule 1 PWM23 Control Select This field selects possible overrides to the generated SM1PWM23 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM1PWM23 signal used by the deadtime logic. 01b - Inverted generated SM1PWM23 signal used by the deadtime logic. 10b - SWCOUT[SM1OUT23] used by the deadtime logic. 11b - PWM1_EXTA signal used by the deadtime logic.
5-4 SM1SEL45	Submodule 1 PWM45 Control Select This field selects possible overrides to the generated SM1PWM45 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM1PWM45 signal used by the deadtime logic. 01b - Inverted generated SM1PWM45 signal used by the deadtime logic. 10b - SWCOUT[SM1OUT45] used by the deadtime logic. 11b - Reserved
3-2 SM0SEL23	Submodule 0 PWM23 Control Select This field selects possible overrides to the generated SM0PWM23 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM0PWM23 signal used by the deadtime logic. 01b - Inverted generated SM0PWM23 signal used by the deadtime logic. 10b - SWCOUT[SM0OUT23] used by the deadtime logic. 11b - PWM0_EXTA signal used by the deadtime logic.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 SM0SEL45	Submodule 0 PWM45 Control Select This field selects possible overrides to the generated SM0PWM45 signal that passes to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM0PWM45 signal used by the deadtime logic. 01b - Inverted generated SM0PWM45 signal used by the deadtime logic. 10b - SWCOUT[SM0OUT45] used by the deadtime logic. 11b - Reserved

54.5.106 Master Control Register (MCTRL)

Offset

Register	Offset
MCTRL	188h

Function

In every 4-bit field in this register, each bit acts on a separate submodule. Accordingly, the description of every bit field refers to the effect of an individual bit.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IPOL				RUN				0				LDOK			
W									CLDOK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-12 IPOL	Current Polarity This field corresponds to submodules 3 - 0, respectively. Each bit selects between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output for the corresponding submodule. MCTRL[IPOL] is ignored in independent mode. MCTRL[IPOL] does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading MCTRL[IPOL] reads the buffered value and not necessarily the value currently in effect. 0000b - PWM23 is used to generate complementary PWM pair in the corresponding submodule. 0001b - PWM45 is used to generate complementary PWM pair in the corresponding submodule.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 RUN	Run This field enables the clocks to the PWM generator of submodules 3 - 0, respectively. The corresponding MCTRL[RUN] bit must be set for each submodule that is using its input capture functions or is using the local reload as its reload source. When this bit equals zero, the submodule counter is reset, and PWM outputs are held. A reset clears this field. 0000b - PWM counter is stopped, but PWM outputs hold the current state. 0001b - PWM counter is started in the corresponding submodule.
7-4 CLDOK	Clear Load Okay This field corresponds to submodules 3 - 0, respectively. Each write-only bit is used to clear the corresponding bit of MCTRL[LDOK]. Write a 1 to CLDOK to clear the corresponding MCTRL[LDOK] bit. If a reload occurs within a submodule with the corresponding MCTRL[LDOK] bit set at the same time that MCTRL[CLDOK] is written, then the reload in that submodule will not be performed and MCTRL[LDOK] will be cleared. CLDOK bit is self-clearing and always reads as a 0.
3-0 LDOK	Load Okay This field corresponds to submodules 3 - 0, respectively. Each read/set bit loads CTRL[PRSC] and the INIT, FRACVALx, and VALx registers of the corresponding submodule into a set of buffers. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if CTRL[LDMOD] is clear or if CTRL[LDMOD] is set. Set the corresponding MCTRL[LDOK] bit by reading it when it is logic zero and then writing a logic one to it. The VALx, FRACVALx, INIT, and CTRL[PRSC] registers of the corresponding submodule cannot be written while the corresponding MCTRL[LDOK] bit is set. In Master Reload Mode (CTRL2[RELOAD_SEL]=1), it is necessary to set the LDOK bit corresponding to submodule0; however, it is recommended to also set the LDOK bit of the slave submodules, to prevent unwanted writes to the registers in the slave submodules. The MCTRL[LDOK] bit is automatically cleared after the new values are loaded, or it can be manually cleared before a reload by writing a logic 1 to the appropriate MCTRL[CLDOK] bit. LDOK bits cannot be written with a zero. MCTRL[LDOK] can be set in DMA mode when the DMA indicates that it has completed the update of all CTRL[PRSC], INIT, FRACVALx, and VALx registers in the corresponding submodule. Reset clears LDOK field. 0000b - Do not load new values. 0001b - Load prescaler, modulus, and PWM values of the corresponding submodule.

54.5.107 Master Control 2 Register (MCTRL2)

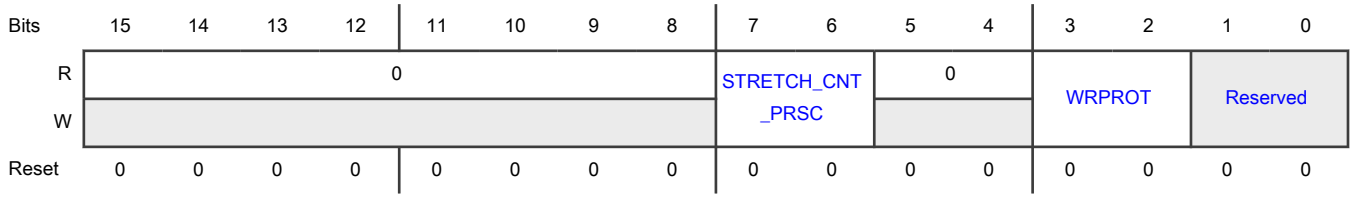
Offset

Register	Offset
MCTRL2	18Ah

Function

Includes control for monitoring the PLL state and write protection of some configuration registers.

Diagram



Fields

Field	Function
15-8 —	Reserved
7-6 STRETCH_CNT_PRSC	Stretch IPBus clock count prescaler for mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig If user config eFlexPWM work in fast clk mode(use SoC level register, eFlexPWM input signal fast_clk_mode is high), then user can use these bits to stretch output signals mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig. 00b - Stretch count is zero, no stretch. 01b - Stretch mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig for 2 IPBus clock period. 10b - Stretch mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig for 4 IPBus clock period. 11b - Stretch mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig for 8 IPBus clock period.
5-4 —	Reserved
3-2 WRPROT	Write protect Enable write protection of some configuration registers of eFlexPWM. 00b - Write protection off (default). 01b - Write protection on. 10b - Write protection off and locked until chip reset. 11b - Write protection on and locked until chip reset.
1-0 —	Reserved

54.5.108 Fault Control Register (FCTRL0)

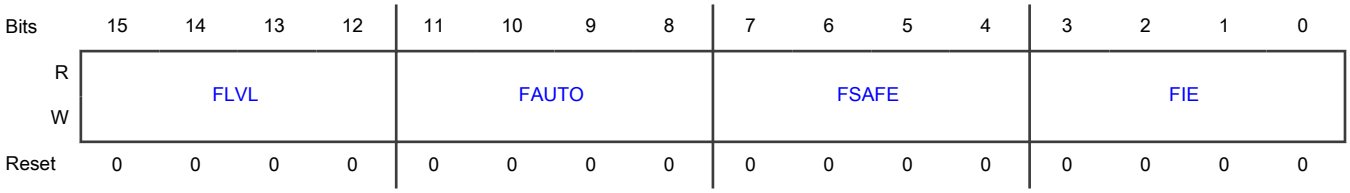
Offset

Register	Offset
FCTRL0	18Ch

Function

For every 4-bit field in this register, the bits act on the fault inputs in order. For example, FLVL bits 15-12 act on faults 3-0, respectively. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-12 FLVL	Fault Level This field selects the active logic level of the individual fault inputs 3-0, respectively. A reset clears this field. 0000b - A logic 0 on the fault input indicates a fault condition. 0001b - A logic 1 on the fault input indicates a fault condition.
11-8 FAUTO	Automatic Fault Clearing This field selects automatic or manual clearing of faults 3-0, respectively. A reset clears this field. 0000b - Manual fault clearing. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared. This is further controlled by FCTRL[FSAFE]. 0001b - Automatic fault clearing. PWM outputs disabled by this fault are enabled when FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFLAGx]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared.
7-4 FSAFE	Fault Safety Mode This field selects the safety mode during manual fault clearing. A reset clears this field. FSTS[FFPINx] may indicate that a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000b - Normal mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFPINx]. If neither FHALF nor FFULL is set, then the fault condition cannot be cleared. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAPn). 0001b - Safe mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear and FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL]. If neither FHLAF nor FFULL is set, then the fault condition cannot be cleared.
3-0 FIE	Fault Interrupt Enables This field enables CPU interrupt requests generated by the FAULTx pins. A reset clears this field. **NOTE** The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register. 0000b - FAULTx CPU interrupt requests disabled. 0001b - FAULTx CPU interrupt requests enabled.

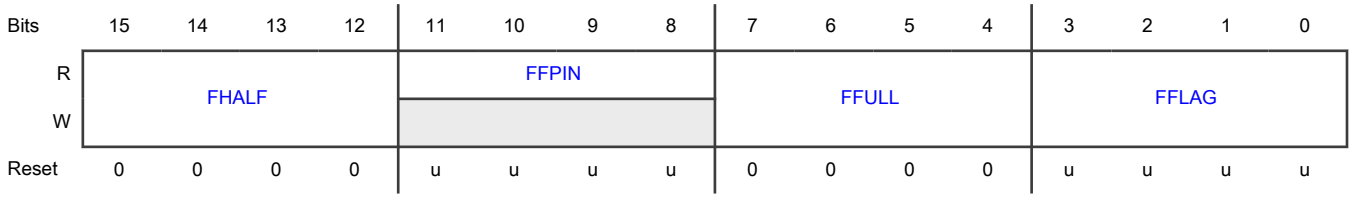
54.5.109 Fault Status Register (FSTS0)

Offset

Register	Offset
FSTS0	18Eh

Function
Includes controls related to fault conditions.

Diagram



Fields

Field	Function
15-12 FHALF	Half Cycle Fault Recovery This field is used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. These register bits are write-protected by MCTRL2[WRPROT] bits. **NOTE** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000b - PWM outputs are not re-enabled at the start of a half cycle. 0001b - PWM outputs are re-enabled at the start of a half cycle (as defined by VAL0).
11-8 FFPIN	Filtered Fault Pins These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates that a fault condition exists on the filtered FAULTx pin. A reset has no effect on this field. After the system reset de-asserts, these are the possible values of FFPIN: If FCTRL[FLVL] = 0 and FAULTx = 0, then FFPIN is set to 1. If FCTRL[FLVL] = 0 and FAULTx = 1, then FFPIN is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 0, then FFPIN is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 1, then FFPIN is set to 1.
7-4 FFULL	Full Cycle This field is used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. These register bits are write-protected by MCTRL2[WRPROT] bits. **NOTE** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000b - PWM outputs are not re-enabled at the start of a full cycle 0001b - PWM outputs are re-enabled at the start of a full cycle
3-0	Fault Flags

Table continues on the next page...

Table continued from the previous page...

Field	Function
FFLAG	These read-only flags are set within two CPU cycles after a transition to active on the FAULTx pin. Clear this bit by writing a logic one to it. A reset clears this field. While the reset value is 0, these bits may be set to 1 by the time they can be read depending on the state of the fault input signals. After the system reset de-asserts, these are the possible values of FFLAG: If FCTRL[FLVL] = 0 and FAULTx = 0, then FFLAG is set to 1. If FCTRL[FLVL] = 0 and FAULTx = 1, then FFLAG is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 0, then FFLAG is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 1, then FFLAG is set to 1. 0000b - No fault on the FAULTx pin. 0001b - Fault on the FAULTx pin.

54.5.110 Fault Filter Register (FFILT0)

Offset

Register	Offset
FFILT0	190h

Function

The settings in this register are shared among each of the fault input filters within the fault channel.

Input filter considerations include:

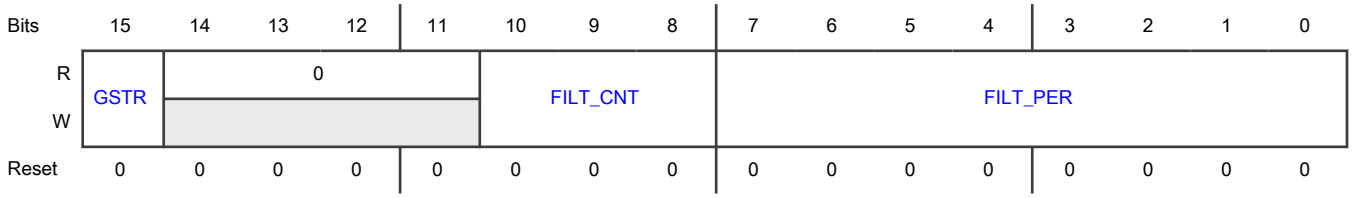
- Set the FILT_PER value such that the sampling period is larger than the period of the expected noise. This way a noise spike corrupts one sample. Select the FILT_CNT value to reduce and recognize the probability of noisy samples causing an incorrect transition. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.
- The values of FILT_PER and FILT_CNT must be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x IPBus clock period).

NOTE

When the filter is enabled, there is a combinational path to disable the PWM outputs to ensure rapid response to fault conditions, and fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15 GSTR	Fault Glitch Stretch Enable This field is used to enable the fault glitch-stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases, a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0b - Fault input glitch stretching is disabled. 1b - Input fault signals are stretched to at least 2 IPBus clock cycles.
14-11 —	Reserved
10-8 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. **NOTE** When changing values for FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

54.5.111 Fault Test Register (FTST0)

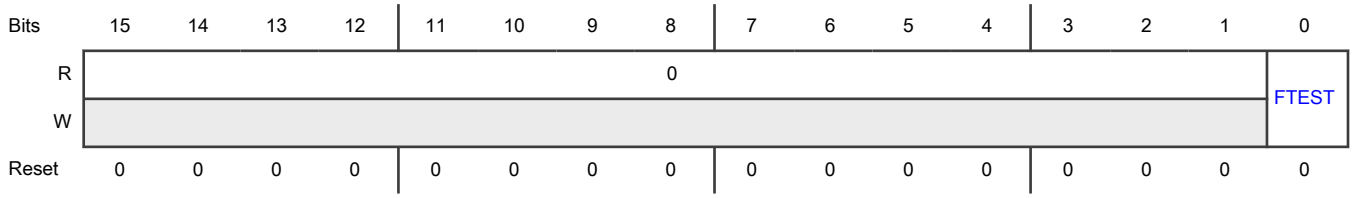
Offset

Register	Offset
FTST0	192h

Function

Contains FTEST field for fault simulation.

Diagram



Fields

Field	Function
15-1 —	Reserved
0 FTEST	Fault Test This field is used to simulate a fault condition. Setting this bit causes a simulated fault to be sent into all of the fault filters. The condition propagates to the fault flags and possibly the PWM outputs depending on the DISMAPn settings. Clearing this bit removes the simulated fault condition. This register bit is write-protected by MCTRL2[WRPROT] bits. 0b - No fault 1b - Cause a simulated fault

54.5.112 Fault Control 2 Register (FCTRL20)

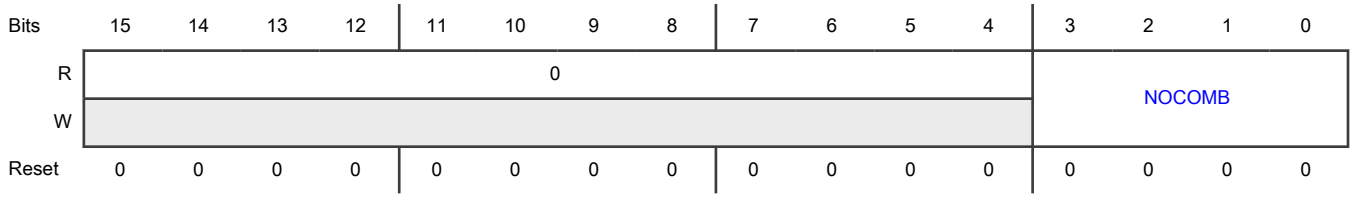
Offset

Register	Offset
FCTRL20	194h

Function

Controls combinational link from fault inputs to PWM outputs. This register is write-protected by MCTRL2[WRPROT] bits.

Diagram



Fields

Field	Function
15-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3-0 NOCOMB	No Combinational Path From Fault Input To PWM Output This field is used to control the combinational path from the fault inputs to the PWM outputs. When these bits are low (default), the corresponding fault inputs have a combinational path to the PWM outputs that are sensitive to these fault inputs (as defined by DISMAP0). This combinational path is a safety feature that ensures the output is disabled even if the SOC has a failure of its clocking system. The combinational path also means that a pulse on the fault input can cause a brief disable of the PWM output even if the fault pulse is not wide enough to get through the input filter and be latched in the fault logic. Setting these bits removes the combinational path and uses the filtered and latched fault signals as the fault source to disable the PWM outputs. This eliminates fault glitches from creating PWM output glitches but also increases the latency to respond to a real fault. 0000b - There is a combinational link from the fault inputs to the PWM outputs. The fault inputs are combined with the filtered and latched fault signals to disable the PWM outputs. 0001b - The direct combinational path from the fault inputs to the PWM outputs is disabled and the filtered and latched fault signals are used to disable the PWM outputs.

Chapter 55
Quadrature Decoder (QDC)

55.1 Chip-specific QDC information

Table 450. Reference links to related information

Topic	Related module	Reference
Full description	QDC	QDC
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Input multiplexing	INPUTMUX	See QDC0_TRIG and QDC1_TRIG registers in INPUTMUX

NOTE

QDC uses system_clk. See QDC0_TRIG and QDC1_TRIG registers in INPUTMUX chapter to see QDC input trigger connections.

55.1.1 Module instances

This device has two instances of the QDC module: QDC0 and QDC1.

55.2 Overview

ENC interfaces to position and speed sensors used in industrial motor control applications. ENC uses the following five input signals from those position and speed sensors to decode shaft position, revolution count, and speed:

- PHASEA
- PHASEB
- INDEX
- TRIGGER
- HOME

55.2.1 Block diagram

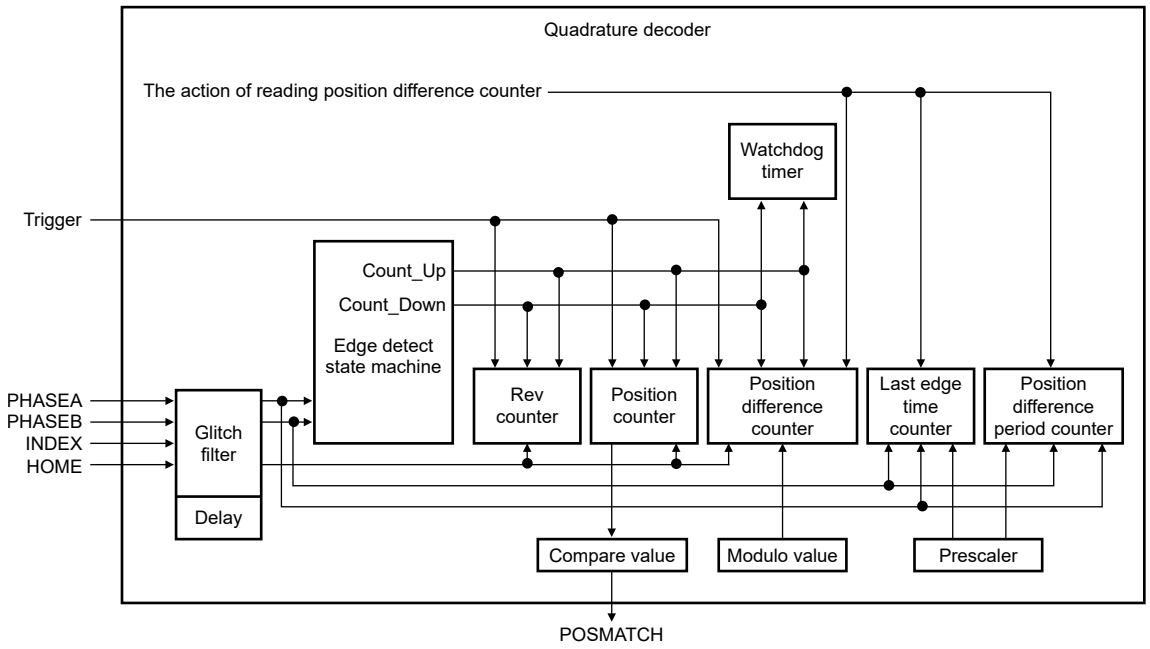


Figure 308. ENC block diagram

55.2.2 Features

- Logic to decode quadrature signals
- Inputs that you could connect to a general purpose timer to perform low-speed velocity measurements
- Digital filter for inputs that you could configure according to your requirements
- Quadrature decoder filter that can be bypassed
- 32-bit position counter capable of modulo counting
- Position counter that you, or external events, could initialize
- 16-bit position difference register
- Compare function to indicate when the shaft has reached a defined position
- Watchdog timer to detect a non-rotating shaft condition
- 16-bit revolution counter that could be preloaded
- Maximum count frequency that equals the peripheral clock rate
- Availability of an optional interrupt when both PHASEA and PHASEB inputs change in the same cycle

55.3 Functional description

The following timing diagram shows the basic operation of an incremental position ENC.

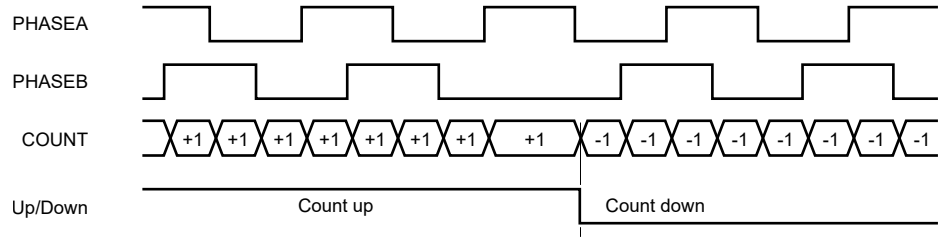


Figure 309. ENC signals

55.3.1 Positive versus negative direction

A typical quad encoder has three outputs: PHASEA signal, PHASEB signal, and INDEX pulse (not shown).

- If PHASEA leads PHASEB, then motion is in the positive direction.
- If PHASEA trails PHASEB, then motion is in the negative direction.

You can integrate the transitions on these phases to yield position or differentiate to yield velocity. Design the ENC to perform these functions in hardware.

55.3.2 Speed measurement method

This section explains the speed measurement method. It includes a speed measurement algorithm.

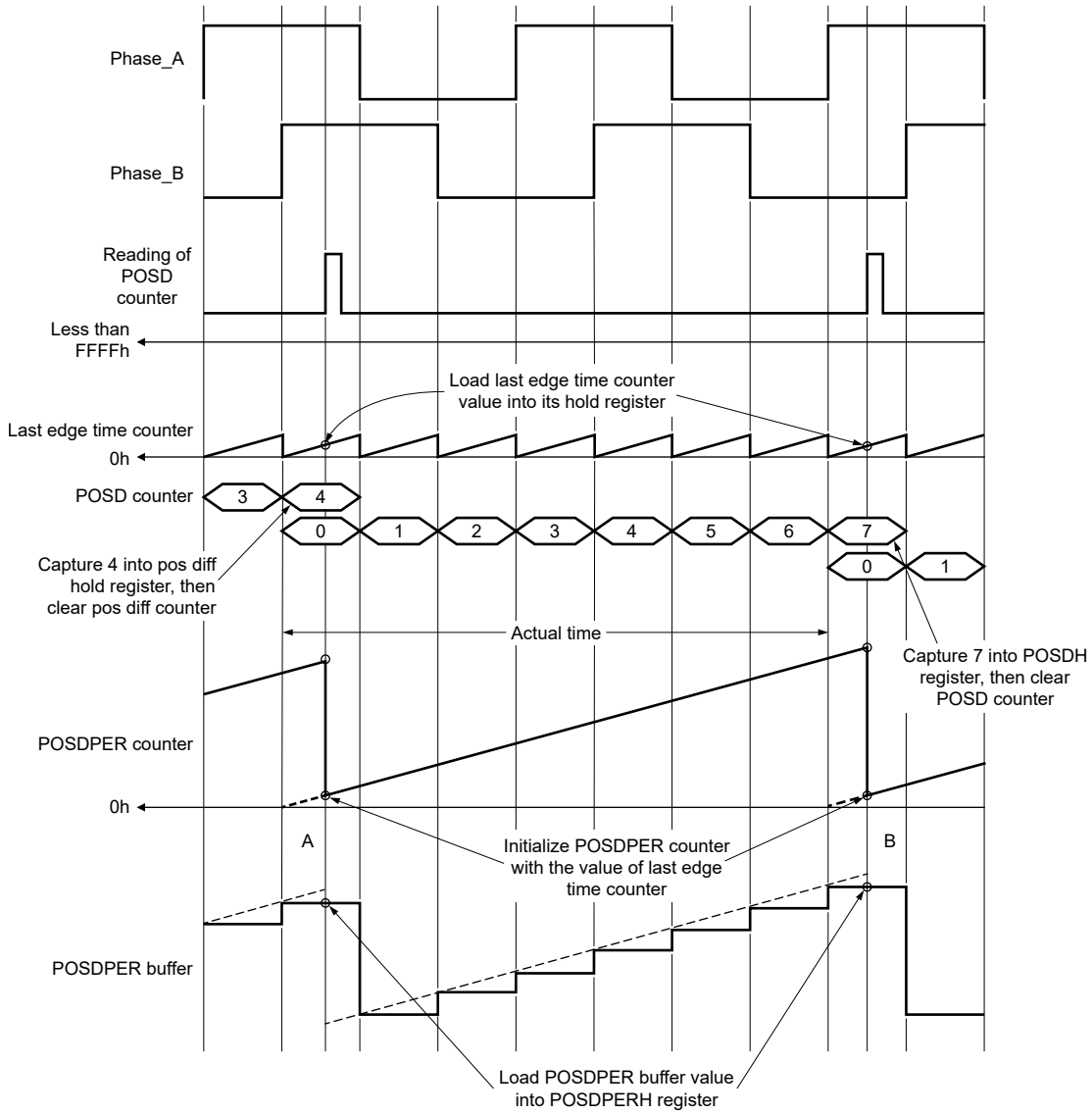


Figure 310. Counters behavior and updating mechanism

Reading of Position Difference Counter (POSD) occurs at time point A and time point B. “Actual time” represents the time duration between the first phase_a/b edges right before time points A and B. At time point B, after reading Position Difference Counter (POSD), Position Difference Period Hold (POSDPERH) contains the time length of “Actual time” and Position Difference Hold (POSDH) includes the value of phase_a/b pulses within that “Actual time”. You can calculate the speed based on Position Difference Period Hold (POSDPERH) and Position Difference Hold (POSDH). The following figure shows a visualized explanation of speed measurement with this method:

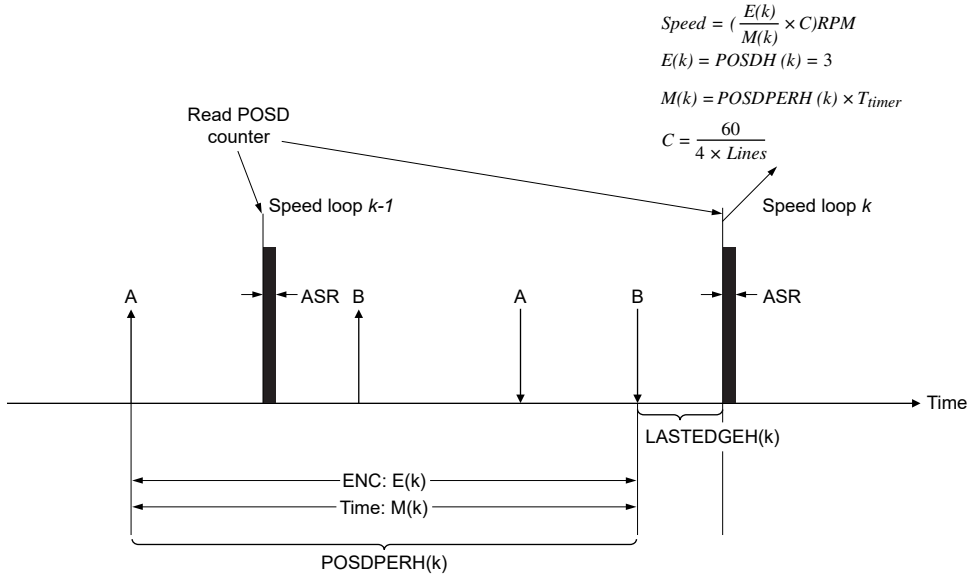
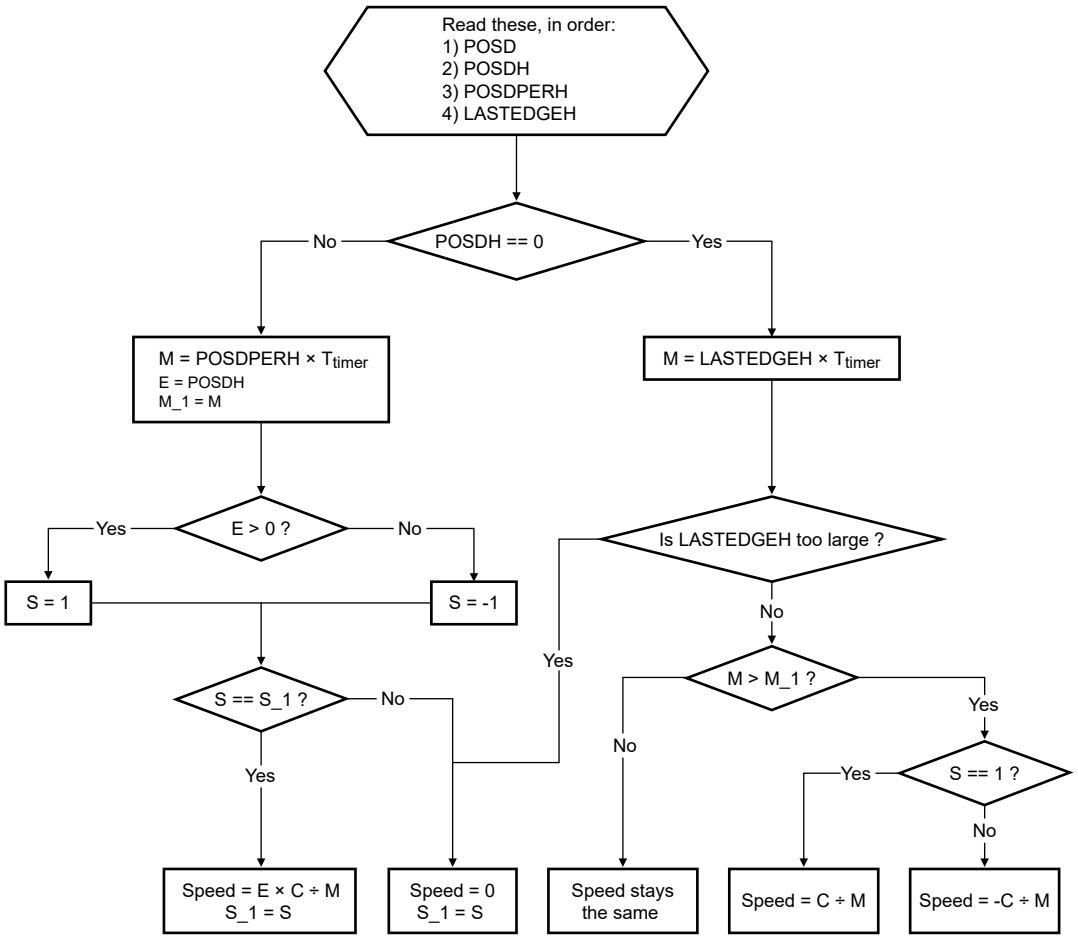


Figure 311. High speed measurement with method

Speed calculation algorithm

However, to cover all the cases in real applications, a simple speed measurement algorithm is necessary. The following figure shows the speed measurement algorithm.



S: Indicates whether E = 0 in this POSD reading operation. If E = 0, then S = -1. If E > 0, then S = 1.
S_1: Indicates whether E = 0 in the previous POSD reading operation. If E=0, then S = -1. If E > 0, then S=1.
M: Indicates the M value in this POSD reading operation.
M_1: Indicates the M value in the previous POSD reading operation.

Figure 312. Flowchart of speed calculation with method

Speed calculation of motor's starting from standstill (Chip is out of reset)

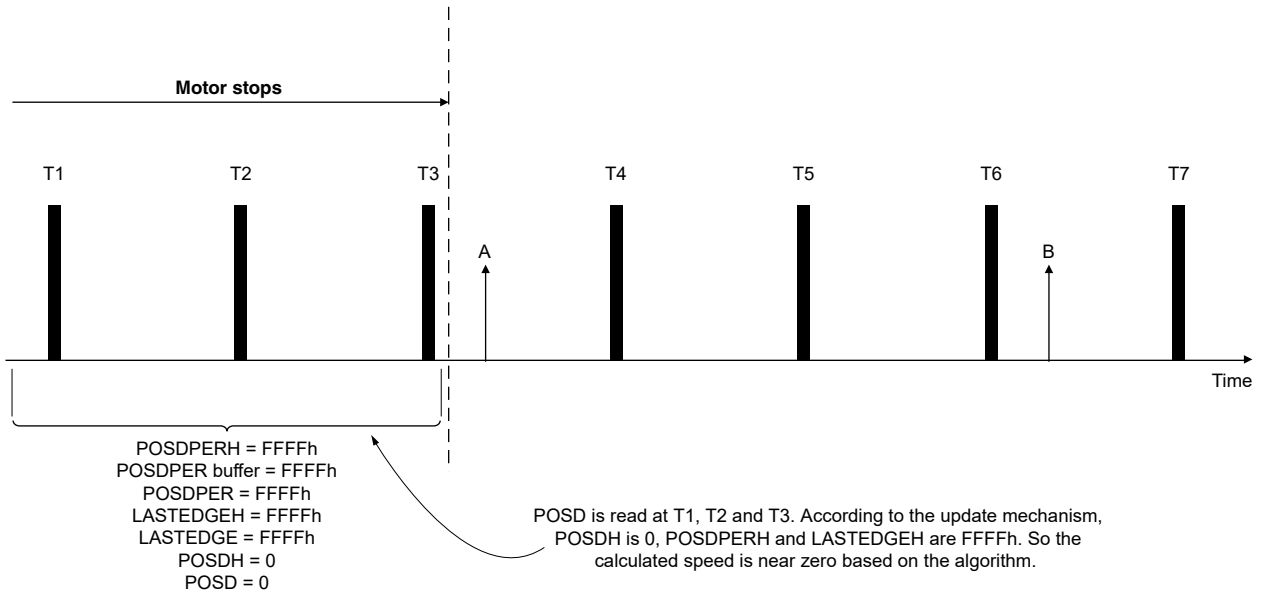


Figure 313. Speed measurement at time point T1~T3

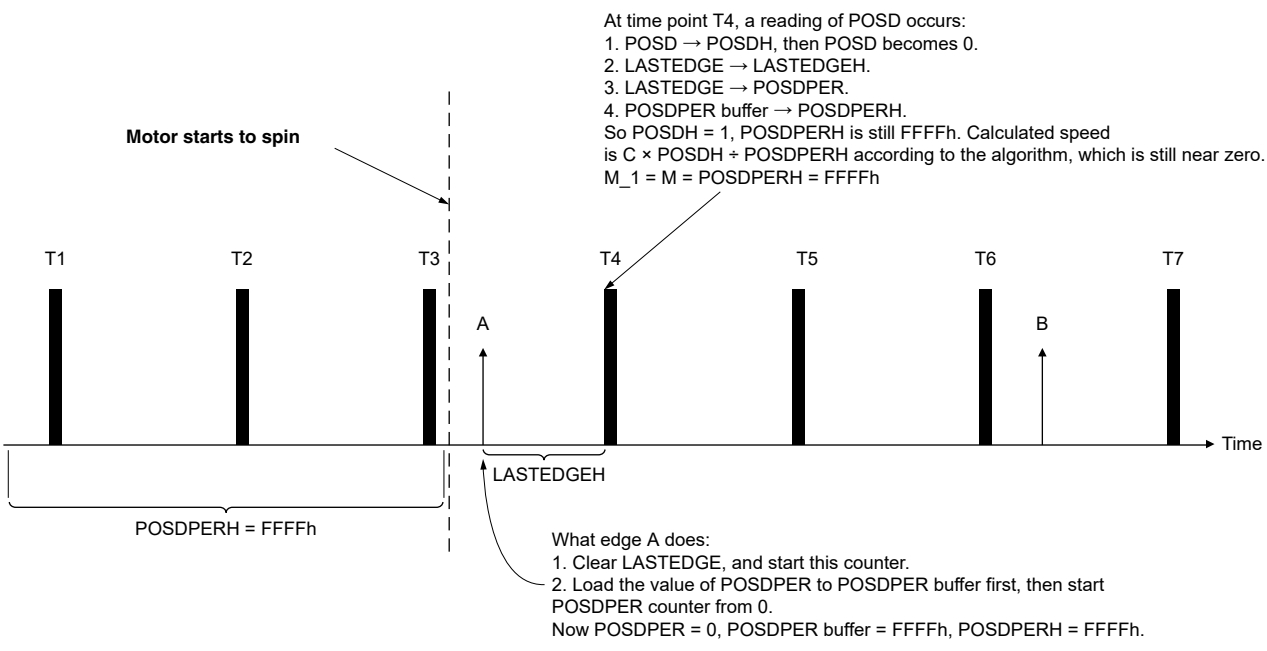


Figure 314. Speed measurement at time point T4

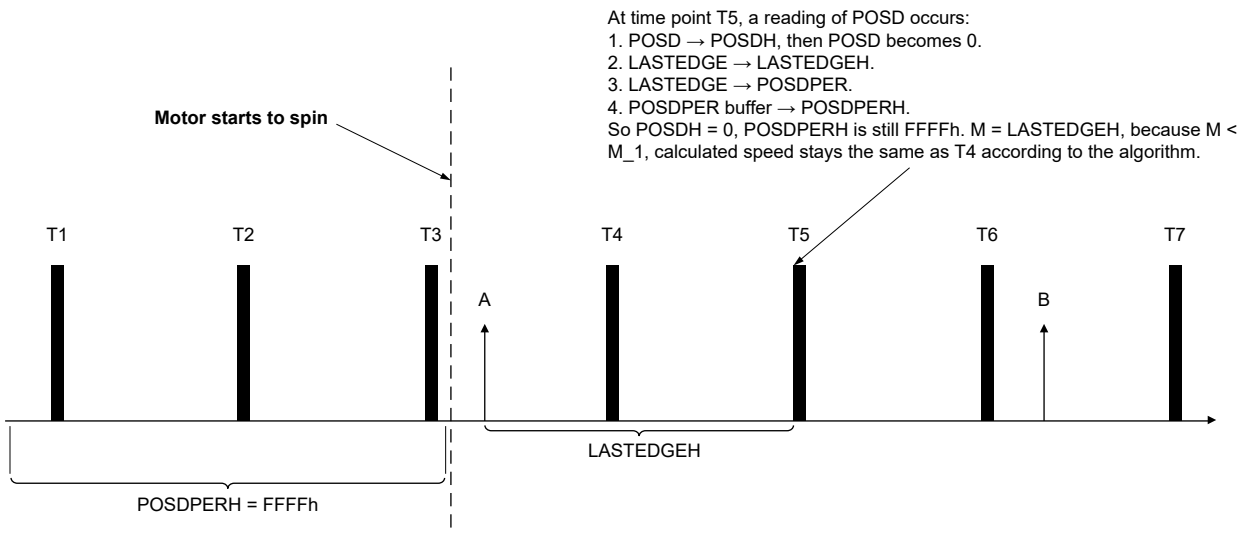


Figure 315. Speed measurement at time point T5

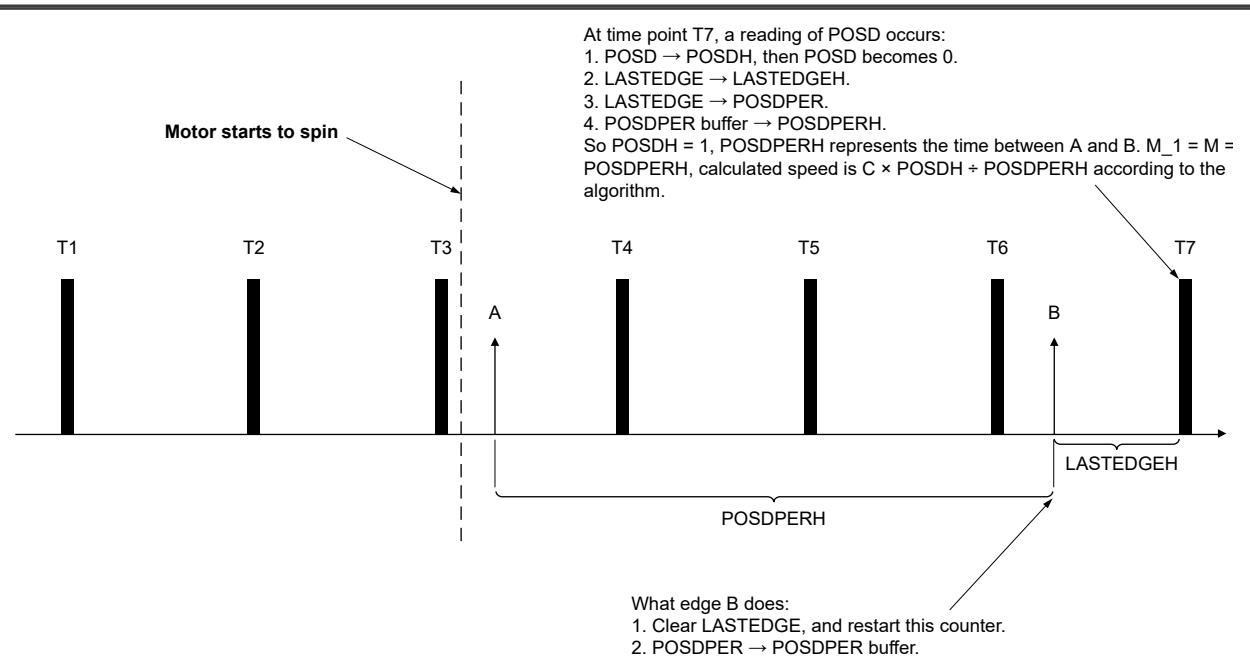


Figure 316. Speed measurement at time point T7

Speed calculation when motor stops

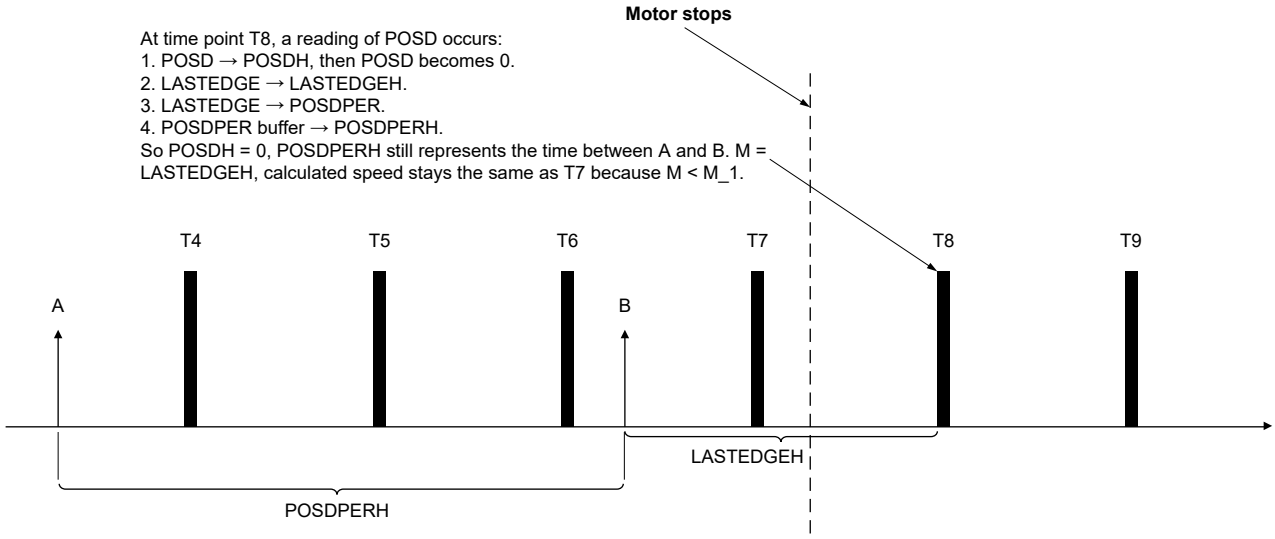


Figure 317. Speed measurement at time point T8

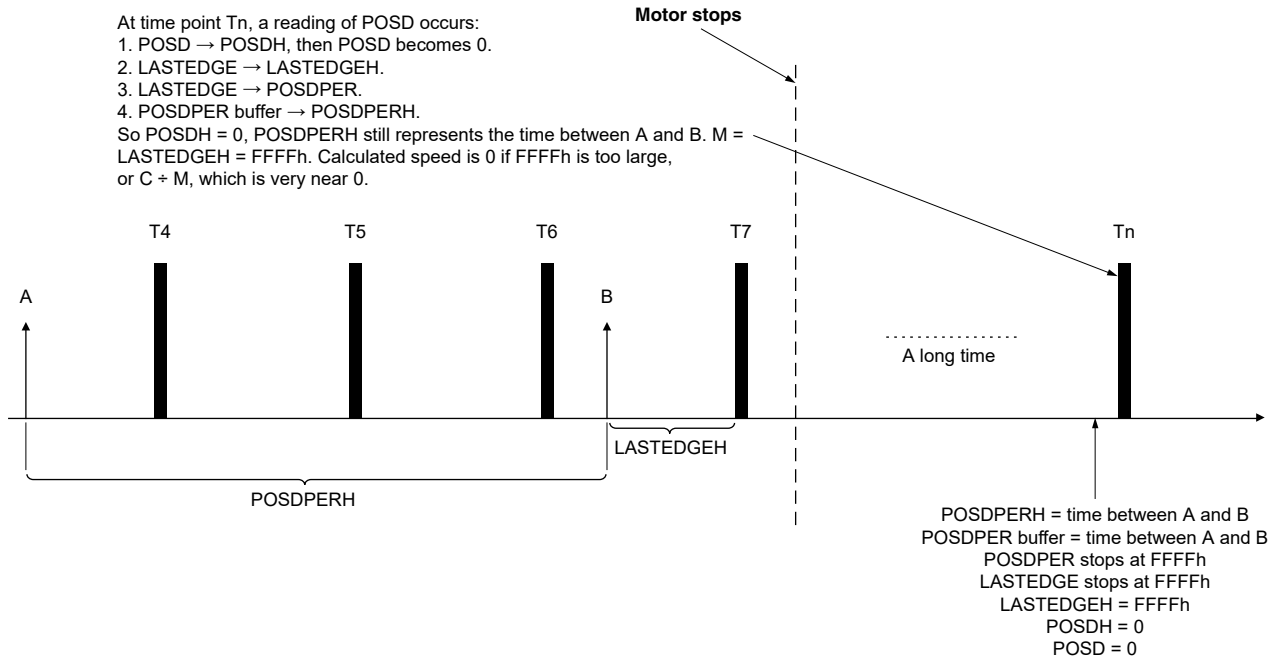


Figure 318. Speed measurement at time point Tn

Speed calculation when motor starts again

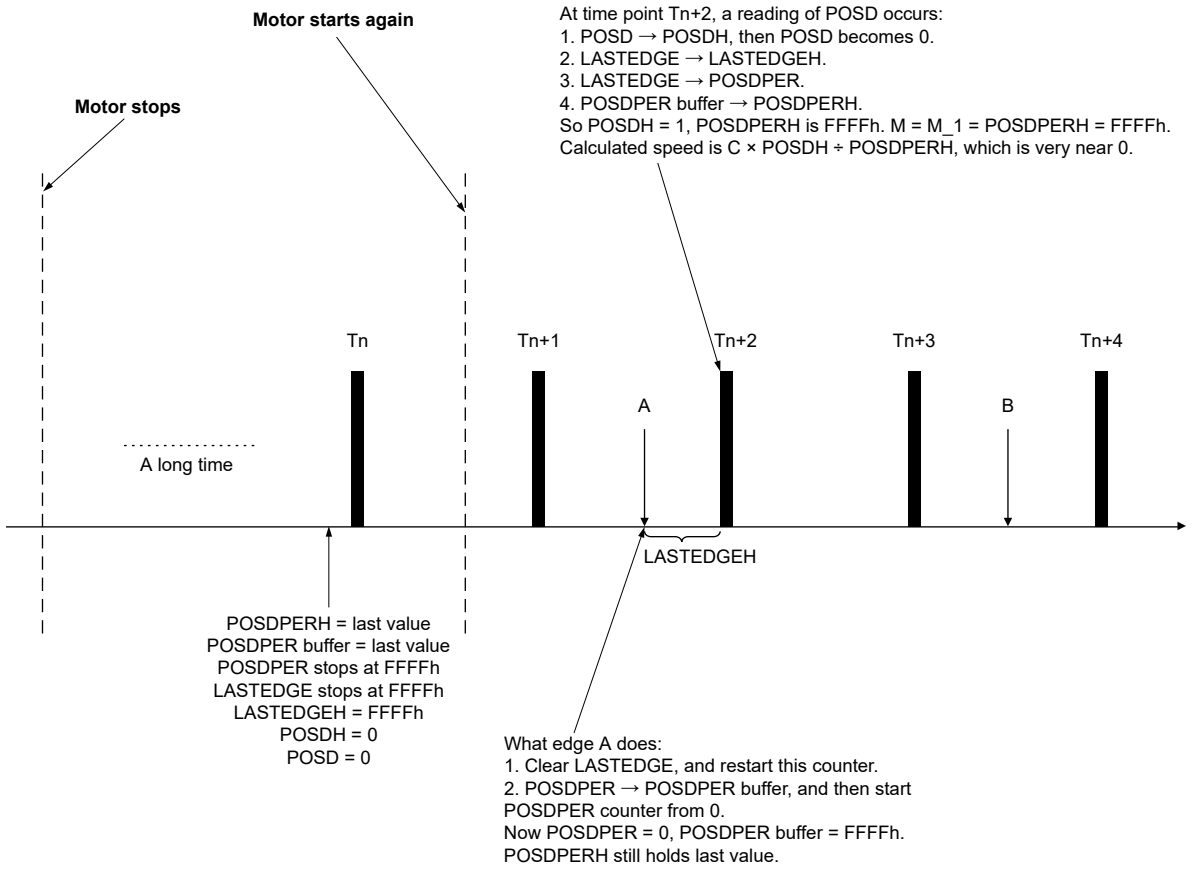


Figure 319. Speed measurement at time point Tn+2

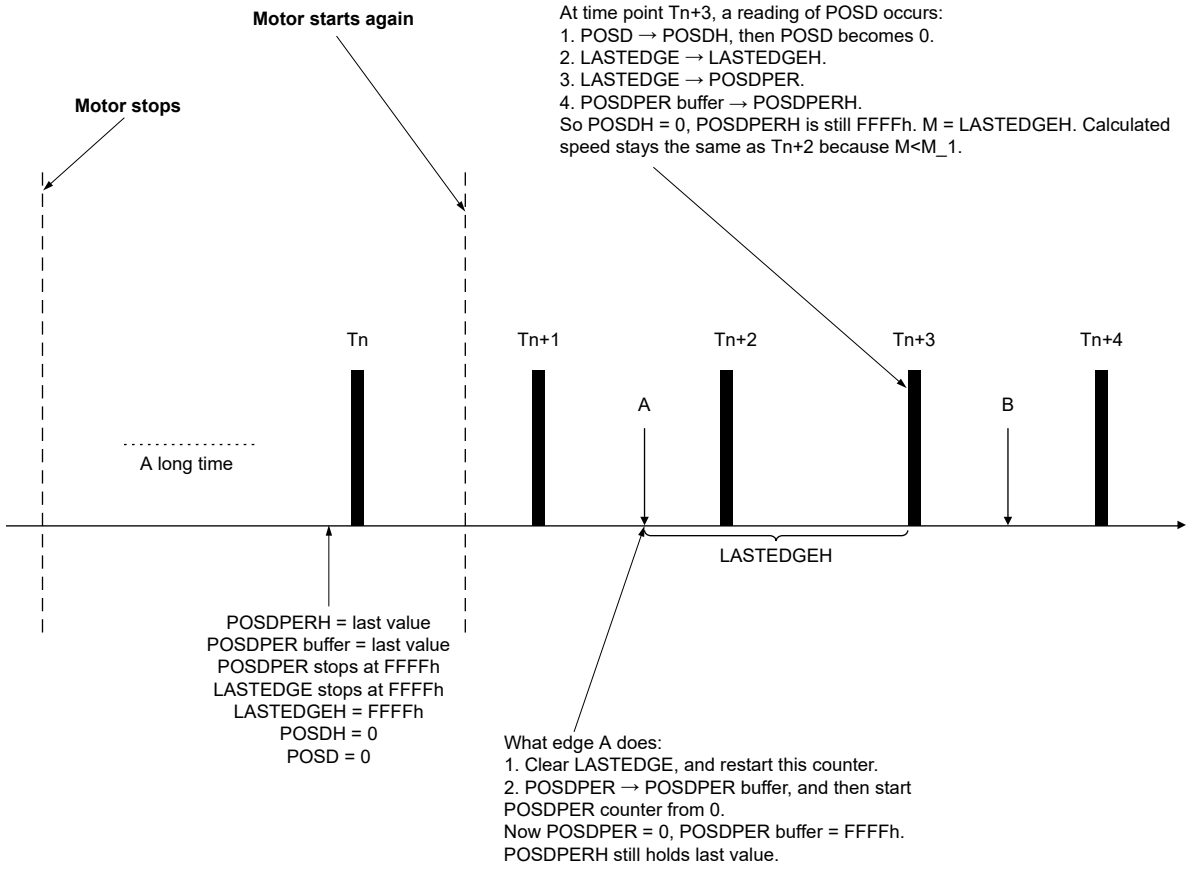


Figure 320. Speed measurement at time point Tn+3

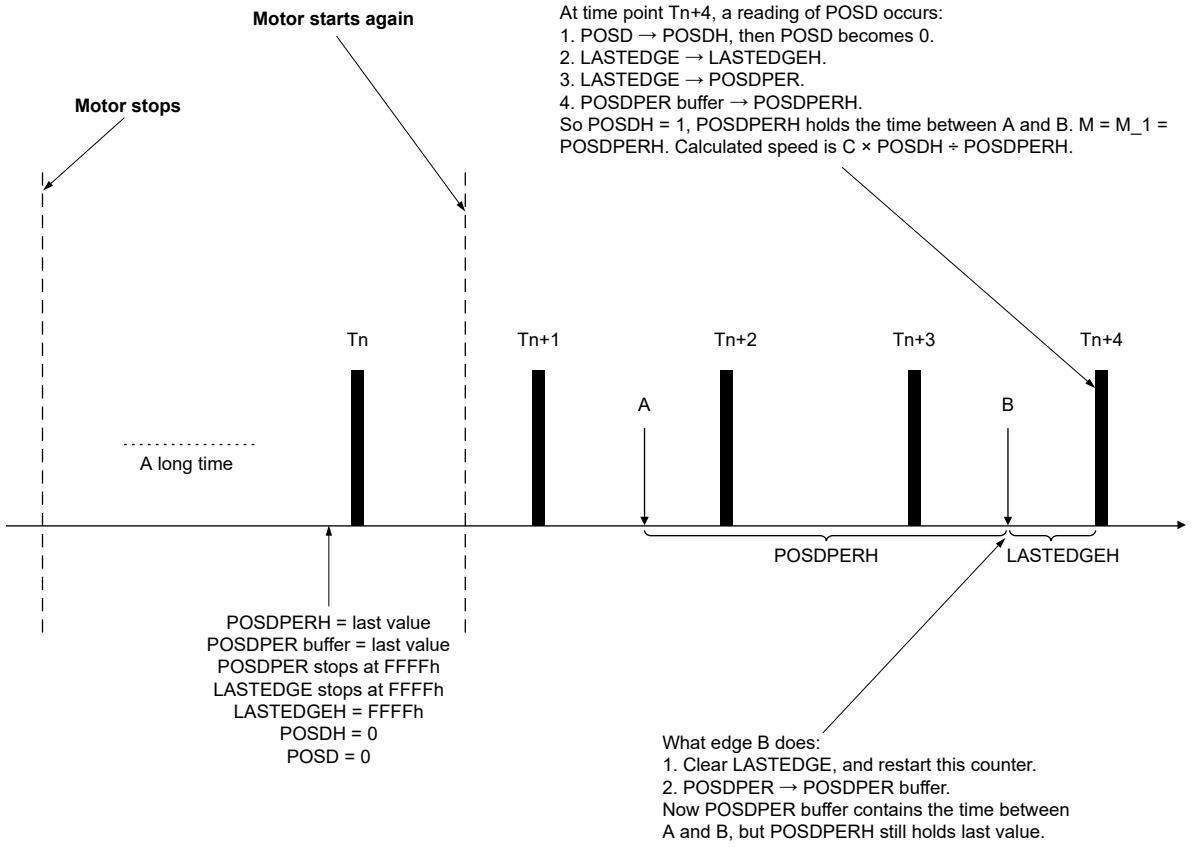


Figure 321. Speed measurement at time point Tn+4

55.3.3 Glitch filter

The quadrature decoder logic must sense signal transitions, and the signal inputs first run through a glitch filter. This glitch filter has a digital delay line, which samples multiple time points on the signal and verifies a stable new signal state before outputting this new signal state to the internal quadrature decoder logic. To adapt to a variety of signal bandwidths, the sample rate of this delay line is programmable.

55.3.4 Edge detect state machine

The edge detect state machine looks for changes in the four possible states of the filtered PHASEA and PHASEB inputs, which calculates the direction of motion. Format this information as Count_Up and Count_Down signals. Each counter includes its hold register. These signals route into up to three up/down counters:

- Position counter
- Revolution counter
- Position difference counter

55.3.5 Counter registers and hold registers, and how to initialize

Hold registers are associated with the following 4 types of counters:

- Position
- Position difference
- Revolution

- Rising edge of trigger event when CTRL2[UPDHLD] is 1

When you read any counter registers, write the contents of each counter register to the corresponding hold register. Taking a snapshot of the counters' values allows a consistent view of a system's position and the velocity to attain. If CTRL3[PMEN] is 1, then Position Difference Hold (POSDH) updates only when you read the POSD counter and not when other counters are read. To capture a time stamp of when these registers are read, use the POSMATCH output with a timer channel.

The position counter is 32 bits wide. To ensure that the position counter can reliably initialize with two 16-bit accesses, two registers (Upper Initialization (UINIT) and Lower Initialization (LINIT)) are provided. You must load Upper Initialization (UINIT) and Lower Initialization (LINIT) with the desired value. Then write 1 to CTRL[SWIP] to load the position counter. Alternatively, CTRL[XIP] or CTRL[HIP] can enable the position counter to initialize in response to a HOME or INDEX signal transition.

55.3.5.1 Position counter

The 32-bit position counter calculates up or down on every count pulse, which the difference between PHASEA and PHASEB generates. The position counter acts as integration information, whose count value is proportional to position. Count_Up and Count_Down signals determine the direction of the count. You may initialize the position counters to a predetermined value by one of 4 different methods:

- Software-triggered event
- INDEX signal transition
- HOME signal transition
- Rising edge of a trigger event

You can program the INDEX and HOME signals to interrupt the processor. Whenever you read the position counters (Upper Position Counter (UPOS) or Lower Position Counter (LPOS)), a snapshot of the following counters are placed into their respective hold registers:

- Position counter
- Position difference counter (only if CTRL3[PMEN] is 0)
- Revolution counter

55.3.5.2 Position difference counter

The 16-bit position difference counter contains the position difference value occurring between each read of the position register. The position register counts up or down on every count pulse. The position difference counter acts as a differentiator whose count value is proportional to the change in position since the last time the position counter was read.

When you read the position registers, the position difference counter, or the revolution counter, the contents of Position Difference Counter (POSD) are copied into Position Difference Hold (POSDH), and the position difference counter becomes 0.

If CTRL3[PMEN] is 1, a read of the position difference counter, and not a read of the revolution counter or position counters, updates the position difference counter and its hold register.

55.3.5.3 Position difference counter hold

Position Difference Hold (POSDH) stores a copy of the position difference counter when you read the position register. When you read the position register, the position difference counter, or the revolution counter, the position difference counter's contents are copied into Position Difference Hold (POSDH), and the position difference counter becomes 0.

If CTRL3[PMEN] is 1, a read of the position difference counter, and not a read of the revolution counter or position counters, updates Position Difference Hold (POSDH).

55.3.5.4 Revolution counter

The 16-bit up/down revolution counter counts or integrate revolutions by counting index pulses. Determine the direction of the count by the Count_Up and Count_Down signals, determined by the Phase A and B inputs. A different count direction on the rising and falling edges of the index pulse indicates that the quad encoder changed direction on the index pulse.

55.3.5.5 Watchdog timer

The watchdog timer ensures that the algorithm indicates motion in the shaft. Two successive counts indicate proper operation and reset the timer.

- The timeout value is programmable.
- When a timeout occurs, you can generate an interrupt to the processor.

55.3.5.6 Pulse accumulator functionality

You can program the logic to integrate only selected transitions of the PHASEA signal. This way, you can use the position counter as a pulse accumulator.

- The count direction is up.
- The INDEX input can optionally initialize the pulse accumulator.

55.3.5.7 Last edge time counter

The 16-bit last edge time counter contains the time since the last edge on PHASEA or PHASEB. The last edge time counter counts up on every prescaled clock pulse.

When you read the position difference counter, copy the content of LASTEDGE[LASTEDGE] into Last Edge Time Hold (LASTEDGEH).

55.3.5.8 Position difference period counter

The position difference period counter:

- Contains the accumulated time from the last time you read the position difference counter.
- Counts up on every prescaled clock pulse.
- Loads from the last edge time counter whenever you read the position difference register.

55.3.6 Prescaler for slow or fast speed measurement

- For applications with a fast-moving shaft encoder, to compute the speed, calculate the change in the position counter per unit time, or read Position Difference Counter (POSD) and calculate the speed.
- The timer module enables high-resolution velocity measurements by measuring the time period between quad phases for applications with slow motor speeds and low line count quad encoders.
  - The timer module uses a 16-bit free running counter operated from a prescaled version of the IPBus clock.
  - The prescaler divides the IPBus clock by values ranging from 1 to 128. A 60 MHz IPBus clock frequency, yields a resolution from 17 ns to 2.1 µs and a maximum count period from 1.09 ms to 140 ms. For example, with a 1000-tooth decoder, you can calculate speeds down to 0.11 rpm using a prescaler.

55.3.7 Resets

There are no special requirements. Any system reset resets this module.

55.3.8 Clocks

This module requires only the IPBus clock in normal operation.

55.3.9 Interrupts

Table 451. Interrupt summary

Core interrupt	Interrupt flag	Interrupt enable	Description
ipi_int_home	CTRL[HIRQ]	CTRL[HIE]	HOME signal transition interrupt
ipi_int_index	CTRL[XIRQ]	CTRL[XIE]	INDEX signal transition interrupt or roll-over/ under interrupt
	CTRL2[ROIRQ]	CTRL2[ROIE]	
	CTRL2[RUIRQ]	CTRL2[RUIE]	
ipi_int_wdog	CTRL[DIRQ]	CTRL[DIE]	Watchdog timeout interrupt
ipi_int_cmp	CTRL[CMPIRQ]	CTRL[CMPIE]	Compare match interrupt
ipi_int_sab	CTRL2[SABIRQ]	CTRL2[SABIE]	Simultaneous PHASEA and PHASEB change interrupt

55.4 External signals

Four external signals are multiplexed to the following four pins of a quad timer module:

- PHASEA
- PHASEB
- INDEX
- HOME

You can connect the following two internal signals to other chip resources:

- TRIGGER
- POSMATCH

Table 452. Signal descriptions

Signal		Description
PHASEA	Phase A input	You can connect the PHASEA input to one of the phases from a two-phase shaft quad encoder output. ENC uses PHASEA and PHASEB to indicate a decoder increment has passed and to calculate its direction. • When you use ENC as a single phase pulse accumulator, you can also use PHASEA as the single input. • The PHASEA input can be an input capture channel for one of the timer modules (in the chip), which you can connect to using a crossbar switch. Direction for two-phase shaft quad encoder output: • PHASEA is the leading phase for a shaft rotating in the positive direction. • PHASEA is the trailing phase for a shaft rotating in the negative direction.
PHASEB	Phase B input	You can connect the PHASEB input to the other phase from a two-phase shaft quad encoder output.

Table continues on the next page...

Table 452. Signal descriptions (continued)

Signal		Description
		• The PHASEB input can be an input capture channel for one of the timer modules (in the chip), which you can connect to using a crossbar switch. Direction for two-phase shaft quad encoder output: • PHASEB is the trailing phase for a shaft rotating in the positive direction. • PHASEB is the leading phase for a shaft rotating in the negative direction.
INDEX	Index input	• Normally connected to the index pulse output of a quad encoder, the INDEX pulse can optionally reset the position counter and the pulse accumulator of ENC. The INDEX pulse also causes a change of state on the revolution counter. Calculate the direction of this state change (increment or decrement) from the PHASEA and PHASEB inputs. • The INDEX input can be an input capture channel for one of the timer modules (in the chip), which you can connect to using a crossbar switch.
HOME	Home switch input	ENC and the timer module can use the HOME input. • You can use the HOME input to trigger the initialization of the position counters (Upper Position Counter (UPOS) and Lower Position Counter (LPOS)). Often the HOME signal is connected to a sensor on the motor or machine, sending a notification that it has reached a defined home position. • You can connect the HOME signal to the timer module (in the chip) using a crossbar switch.
TRIGGER	Trigger input	You can use the TRIGGER input to: • Write 0 to the position counters (Upper Position Counter (UPOS) and Lower Position Counter (LPOS)). • Take a snapshot of POS, Revolution Counter (REV), and Position Difference Counter (POSD). • Indicate an elapsed time period by connecting the TRIGGER input signal to a periodic pulse generator or timer.
POSMATCH	Position match output	• To record the time at which the shaft position matches a user-defined compare value (COMP), The POSMATCH output can trigger a timer channel. • Alternatively, to record the time at which you read the position information (when you read the position counters (Upper Position Counter (UPOS) and Lower Position Counter (LPOS), Revolution Counter (REV), or Position Difference Counter (POSD)) the POSMATCH output can trigger a timer channel.

55.5 Memory map and registers

This section describes the module memory map and registers.

55.5.1 ENC register descriptions

The address of a register is the sum of a base address and an address offset.

- Base address is defined at the MCU level
- Address offset is defined at the module level

For the base address, see the chip-specific documentation. All memory locations base and offsets are given in hex.

55.5.1.1 ENC memory map

QDC0 base address: 400C_F000h

QDC1 base address: 400D_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CTRL)	16	RW	0000h
2h	Input Filter (FILT)	16	RW	0000h
4h	Watchdog Timeout (WTR)	16	RW	0000h
6h	Position Difference Counter (POSD)	16	RW	0000h
8h	Position Difference Hold (POSDH)	16	R	0000h
Ah	Revolution Counter (REV)	16	RW	0000h
Ch	Revolution Hold (REVH)	16	R	0000h
Eh	Upper Position Counter (UPOS)	16	RW	0000h
10h	Lower Position Counter (LPOS)	16	RW	0000h
12h	Upper Position Hold (UPOSH)	16	R	0000h
14h	Lower Position Hold (LPOSH)	16	R	0000h
16h	Upper Initialization (UINIT)	16	RW	0000h
18h	Lower Initialization (LINIT)	16	RW	0000h
1Ah	Input Monitor (IMR)	16	R	0000h
1Ch	Test (TST)	16	RW	0000h
1Eh	Control 2 (CTRL2)	16	RW	0000h
20h	Upper Modulus (UMOD)	16	RW	0000h
22h	Lower Modulus (LMOD)	16	RW	0000h
24h	Upper Position Compare (UCOMP)	16	RW	FFFFh
26h	Lower Position Compare (LCOMP)	16	RW	FFFFh
28h	Last Edge Time (LASTEDGE)	16	R	FFFFh
2Ah	Last Edge Time Hold (LASTEDGEH)	16	R	FFFFh
2Ch	Position Difference Period Counter (POSDPER)	16	R	FFFFh
2Eh	Position Difference Period Buffer (POSDPERBFR)	16	R	FFFFh
30h	Position Difference Period Hold (POSDPERH)	16	R	FFFFh
32h	Control 3 (CTRL3)	16	RW	0000h

55.5.1.2 Control (CTRL)

Offset

Register	Offset
CTRL	0h

Function

Controls basic functions on HOME and INDEX signals and part of basic functions.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	CMPIRQ	CMPIE
W	W1C				SWIP			W1C				W1C			W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15 HIRQ	HOME Signal Transition Interrupt Request Specifies whether a transition has occurred on the HOME signal. This field becomes 1 when a HOME signal transitions, according to CTRL[HNE]. If this field and CTRL[HIE] are 1, then a HOME interrupt occurs. 0b - Not occurred 1b - Occurred
14 HIE	HOME Interrupt Enable Enables or disables HOME signal interrupts. 0b - Disable 1b - Enable
13 HIP	Enable HOME to Initialize Position Counters UPOS and LPOS Enables the position counter, which the HOME signal initializes. 0b - No action 1b - HOME signal initializes the position counter
12 HNE	Use Negative Edge of HOME Input Selects using the positive and negative edge of the HOME input. 0b - Use positive-going edge-to-trigger initialization of position counters UPOS and LPOS 1b - Use negative-going edge-to-trigger initialization of position counters UPOS and LPOS

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 SWIP	Software-Triggered Initialization of Position Counters UPOS and LPOS Specifies that writing 1 to this field transfers the Upper Initialization (UINIT) and Lower Initialization (LINIT) contents to Upper Position Counter (UPOS) and Lower Position Counter (LPOS) (thereby initializing those counters). 0b - No action 1b - Initialize position counter
10 REV	Enable Reverse Direction Counting Selects the direction of the count (how you interpret the quadrature signal). 0b - Counts normally 1b - Counts in the reverse direction
9 PH1	Enable Signal Phase Count Mode Bypasses or uses the quadrature decoder logic. 0b - Uses the standard quadrature decoder, where PHASEA and PHASEB represent a two-phase quadrature signal. 1b - Bypasses the quadrature decoder. A positive transition of the PHASEA input generates a count signal. PHASEB input and CTRL[REV] controls the counter direction. If the value of CTRL[REV] and PHASEB are identical; then count is up. If the value of CTRL[REV] and PHASEB is different, then count is down.
8 XIRQ	INDEX Pulse Interrupt Request Specifies whether the INDEX pulse has occurred. This field becomes 1 when a transition on the INDEX signal occurs, according to the use negative edge of CTRL[XNE]. If this field and CTRL[XIE] become 1, then an INDEX interrupt occurs. 0b - Not occurred 1b - Occurred
7 XIE	INDEX Pulse Interrupt Enable Enables or disables INDEX pulse interrupts. 0b - Disable 1b - Enable
6 XIP	INDEX Triggered Initialization of Position Counters UPOS and LPOS Enables or disables the position counter that the INDEX pulse initializes. 0b - Does not initialize 1b - Initializes
5 XNE	Select Positive and Negative Edge of INDEX Pulse Selects the positive and negative edge of the INDEX pulse.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Use positive edge 1b - Use negative edge
4 DIRQ	Watchdog Timeout Interrupt Request Specifies whether the watchdog timeout interrupt has occurred. This field also becomes 0 when CTRL[WDE] = 0. 0b - Not occurred 1b - Occurred
3 DIE	Watchdog Timeout Interrupt Enable Enables or disables the watchdog timeout interrupts. 0b - Disable 1b - Enable
2 WDE	Watchdog Enable Enables or disables the watchdog timer that monitors the PHASEA and PHASEB inputs for motor movement. 0b - Disable 1b - Enable
1 CMPIRQ	Compare Interrupt Request Specifies whether the COMP match has occurred. This field becomes 1 when the counter matches the COMP value. 0b - No match has occurred 1b - COMP match has occurred
0 CMPIE	Compare Interrupt Enable Enables the compare interrupt. 0b - Disable 1b - Enable

55.5.1.3 Input Filter (FILT)

Offset

Register	Offset
FILT	2h

Function

Specifies:

- The input filter sample period.
- The filter prescaler.
- The filter sample count.

FILT[FILT_PRSC] prescales the IPBus clock according to the following equation:

$$FILT\ clock\ frequency = \frac{IPBus\ clock\ frequency}{2^{FILT_PRSC}}$$

Figure 322. FILT clock frequency

Choose FILT[FILT_PER] such that the sampling period is larger than the period of the expected noise. Doing this means that a noise spike will only corrupt one sample.

Choose FILT[FILT_CNT] to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of FILT_CNT+3.

You must also trade off the values of FILT[FILT_PER] and FILT[FILT_CNT] against the need for minimal latency in recognizing valid input transitions. Turning on the input filter (write a non-zero value to FILT[FILT_PER]) introduces a latency of ((FILT[FILT_CNT] + 3)×FILT[FILT_PER]) FILT clock cycles + 2 IPBus clock periods.

You can measure the filter latency:

- Drive the quadrature decoder inputs (PHASEA, PHASEB, INDEX, HOME).
- Monitor the filtered output in Input Monitor (IMR).
- Determine how many IPBus clock cycles it takes before the output shows up, using the following equations, where f is FILT[FILT_PER], and s is FILT[FILT_CNT].
  - DELAY = f×(s+3)(FILT clock cycles) + 1 (IPBus clock cycles)(to read the filtered output)
  - DELAY = f×(s+3)(FILT clock cycles) + 2 (IPBus clock cycles) (to monitor the output in the IMR)

One additional IPBus clock cycle is required to read the filtered output, and two more IPBus clock cycles are required to monitor the filtered output in Input Monitor (IMR). The sample rate sets when it reaches the number f. The following examples use the preceding equations:

- Example: when f = 0, the filter is bypassed: DELAY = 1 or 2 clock cycles.
- Example: when f = 5 and s = 2: DELAY = 5×(2+3) FILT clock cycles + (1 or 2) IPBus clock cycles

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FILT_PRSC			0		FILT_CNT			FILT_PER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13	Prescaler Divide IPBus Clock to FILT Clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
FILT_PRSC	Prescales the IPBus clock to FILT clock. A value of 2^{FILT_PRSC} prescales the IPBus clock, which means that the prescaler logic can divide the clock by a minimum of 1 and a maximum of 128. So FILTER can filter lower frequency noise.
12-11 —	Reserved
10-8 FILT_CNT	Input Filter Sample Count Represents the number of consecutive samples that must agree, before the input filter accepts an input transition. • A value of 0h represents three samples • A value of 7h represents ten samples The value of this field affects the input latency.
7-0 FILT_PER	Input Filter Sample Period Represents the decoder input signals' sampling period (in IPBus clock cycles). Sample each input multiple times at the rate this field specifies. • If FILT_PER is 00h, then the input filter is bypassed. Bypassing the digital filter enables the position/ position difference counters to operate with count rates up to the IPBus frequency. • The value of this field affects the input latency. • When you change the value of this field from a non-zero value to another non-zero value, write 0 first to clear the filter.

55.5.1.4 Watchdog Timeout (WTR)

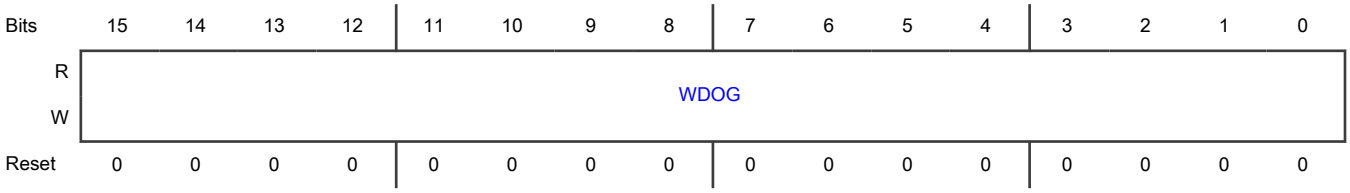
Offset

Register	Offset
WTR	4h

Function

Stores the timeout count for the quadrature decoder module watchdog timer. This quadrature decoder module watchdog timer is different from any other watchdog timer(s) that may also be present in the chip.

Diagram



Fields

Field	Function
15-0 WDOG	WDOG Represents the number of clock cycles, plus one clock cycle that the watchdog timer counts before timing out (and optionally generating an interrupt).

55.5.1.5 Position Difference Counter (POSD)

Offset

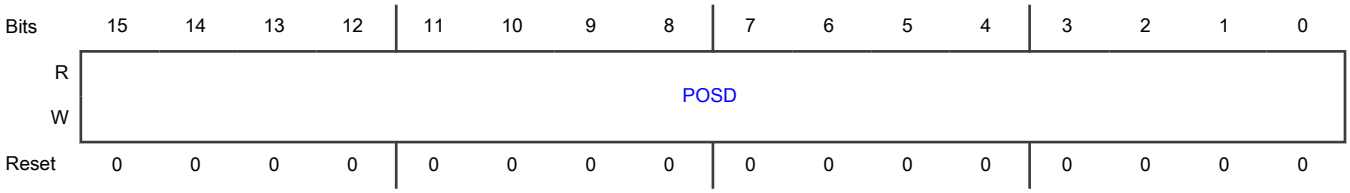
Register	Offset
POSD	6h

Function

Contains the position change in value occurring between each read of the position register. You can use the value of this register to calculate velocity.

- This register computes up or down on every count pulse.
- This register acts as a differentiator, whose count value is proportional to the change in position since the last time you read the position counter.
- When you read this register, its contents are copied into Position Difference Hold (POSDH) and this register becomes 0.

Diagram



Fields

Field	Function
15-0 POSD	POSD Specifies the position change in value between each read of the position register.

55.5.1.6 Position Difference Hold (POSDH)

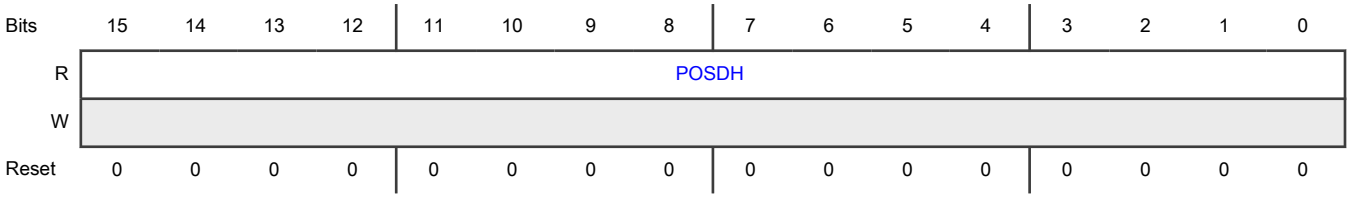
Offset

Register	Offset
POSDH	8h

Function

Contains a snapshot of the value of Position Difference Counter (POSD). You can use the value of this register to calculate velocity.

Diagram



Fields

Field	Function
15-0 POSDH	POSDH Holds the Position Difference Counter (POSD) value.

55.5.1.7 Revolution Counter (REV)

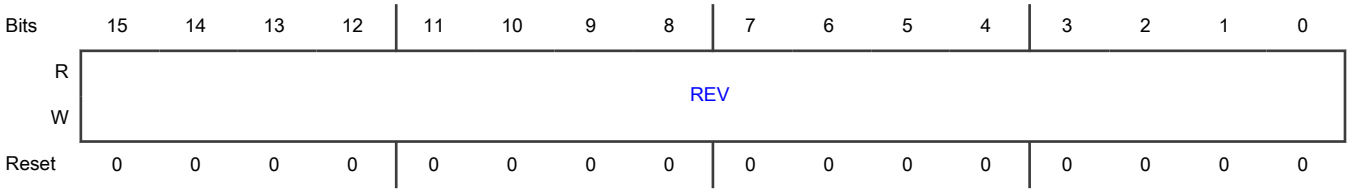
Offset

Register	Offset
REV	Ah

Function

Contains the current value of the revolution counter.

Diagram



Fields

Field	Function
15-0 REV	REV Contains the current value of the revolution counter.

55.5.1.8 Revolution Hold (REVH)

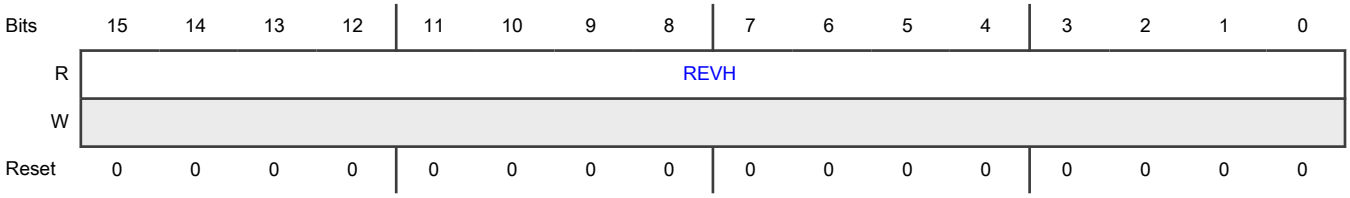
Offset

Register	Offset
REVH	Ch

Function

Contains a snapshot of the value of Revolution Counter (REV).

Diagram



Fields

Field	Function
15-0 REVH	REVH Holds the Revolution Counter (REV) value.

55.5.1.9 Upper Position Counter (UPOS)

Offset

Register	Offset
UPOS	Eh

Function

Contains the upper (most significant) half of the position counter. This is the binary count from the position counter.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-0 POS	POS Contains the upper (most significant) half of the position counter.

55.5.1.10 Lower Position Counter (LPOS)

Offset

Register	Offset
LPOS	10h

Function

Contains the lower (least significant) half of the position counter. This is the binary count from the position counter.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-0 POS	POS Contains the lower (least significant) half of the position counter.

55.5.1.11 Upper Position Hold (UPOSH)

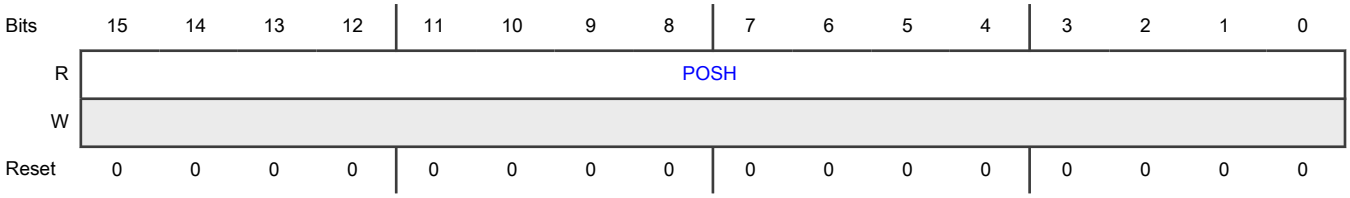
Offset

Register	Offset
UPOSH	12h

Function

Contains a snapshot of Upper Position Counter (UPOS).

Diagram



Fields

Field	Function
15-0 POSH	POSH Holds the Upper Position Counter (UPOS) value.

55.5.1.12 Lower Position Hold (LPOSH)

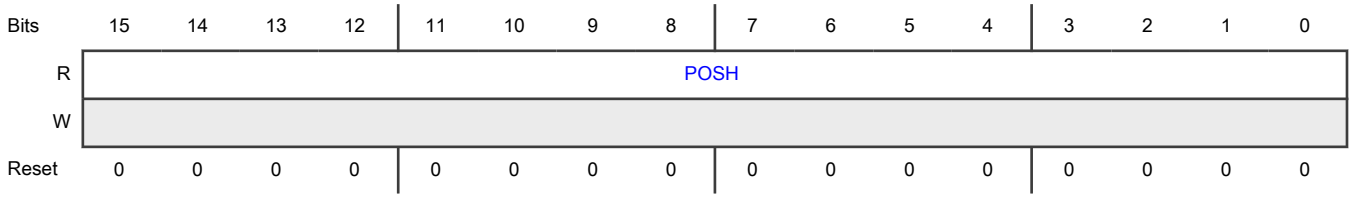
Offset

Register	Offset
LPOSH	14h

Function

Contains a snapshot of Lower Position Counter (LPOS).

Diagram



Fields

Field	Function
15-0 POSH	POSH Holds the Lower Position Counter (LPOS) value.

55.5.1.13 Upper Initialization (UINIT)

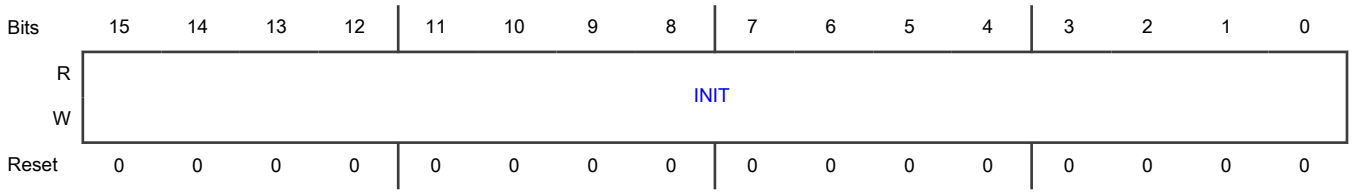
Offset

Register	Offset
UINIT	16h

Function

Contains the value to initialize Upper Position Counter (UPOS).

Diagram



Fields

Field	Function
15-0 INIT	INIT Contains the value to initialize Upper Position Counter (UPOS).

55.5.1.14 Lower Initialization (LINIT)

Offset

Register	Offset
LINIT	18h

Function

Contains the value to initialize Lower Position Counter (LPOS).

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INIT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-0 INIT	INIT Contains the value to initialize Lower Position Counter (LPOS).

55.5.1.15 Input Monitor (IMR)

Offset

Register	Offset
IMR	1Ah

Function

Contains the values of the raw and filtered PHASEA, PHASEB, INDEX, and HOME input signals. The reset value depends on the values of the raw and filtered values of PHASEA, PHASEB, INDEX, and HOME input signals:

- If you connect these input pins to a pullup, then bits 0–7 of this register are all ones.
- If you connect these input pins to a pulldown device, then bits 0–7 of this register are all zeros.
- If no pullup or pulldown is connected to these input pins, then the reset value of the 8 lower bits of this register are all unknown.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								FPHA	FPHB	FIND	FHOM	PHA	PHB	INDEX	HOME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7 FPHA	FPHA Specifies the filtered version of PHASEA input.
6 FPHB	FPHB Specifies the filtered version of PHASEB input.
5 FIND	FIND Specifies the filtered version of INDEX input.
4 FHOM	FHOM Specifies the filtered version of HOME input.
3 PHA	PHA Specifies the raw PHASEA input.
2 PHB	PHB Specifies the raw PHASEB input.
1 INDEX	INDEX Specifies the raw INDEX input.
0 HOME	HOME Specifies the raw HOME input.

55.5.1.16 Test (TST)

Offset

Register	Offset
TST	1Ch

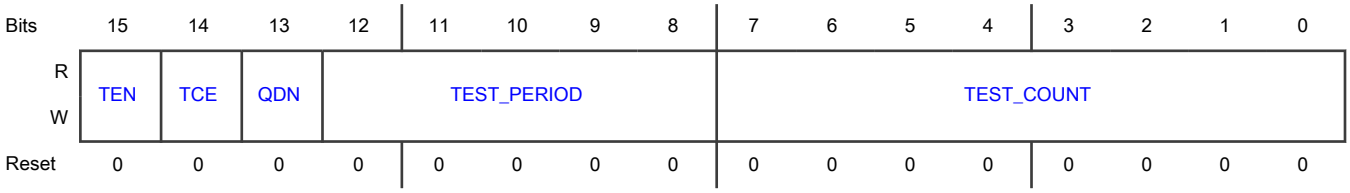
Function

Controls and sets the frequency of a quadrature signal generator; it provides a quadrature test signal to the inputs of the quadrature decoder module.

- The TEST_COUNT value is counted down to zero when TST[TEN] and TST[TCE] are 1.
- Each count value of one represents a single quadrature cycle, which the position counter (Upper Position Counter (UPOS) and Lower Position Counter (LPOS)) interprets as a count of one if enabled (TST[TEN] and TST[TCE] are 1).
- Repeated writing of new values to TEST_COUNT can cause an extra phase transition and, therefore an extra count by the Position Counter.
- The period field determines each quadrature cycle phase's length (in IPBus clock cycles).

This register is a factory test feature; however, you may find it helpful during your software development and testing.

Diagram



Fields

Field	Function
15 TEN	Test Mode Enable Connects the test module to inputs of the quadrature decoder module. 0b - Disable 1b - Enable
14 TCE	Test Counter Enable Connects the test counter to inputs of the quadrature decoder module. 0b - Disable 1b - Enable
13 QDN	Quadrature Decoder Negative Signal Selects whether a negative or positive quadrature decoder signal is generated. 0b - Positive quadrature decoder signal 1b - Negative quadrature decoder signal
12-8 TEST_PERIOD	TEST_PERIOD Specifies the period of the quadrature phase in IPBus clock cycles.
7-0 TEST_COUNT	TEST_COUNT Specifies the number of quadrature advances to generate.

55.5.1.17 Control 2 (CTRL2)

Offset

Register	Offset
CTRL2	1Eh

Function

Specifies the status of interrupts, interrupt enable, and part of basic functions.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			INITPOS	SABIRQ	SABIE	OUTCTL	REVMOD	ROIRQ	ROIE	RUIRQ	RUIE	DIR	MOD	UPDPOS	UPDHLD
W					W1C				W1C		W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-13 —	Reserved
12 INITPOS	Initialize Position Registers Has multiple functions: • When this field is 1, it allows the TRIGGER input to initialize the position counter with Upper Initialization (UINIT) and Lower Initialization (LINIT). • When this field is 0, Upper Position Counter (UPOS) and Lower Position Counter (LPOS) are not initialized on the rising edge of TRIGGER input. Do not write 1 to this field when you use Last Edge Time (LASTEDGE) and Position Difference Period Counter (POSDPER) to measure speed. 0b - Don't initialize position counter 1b - Initialize position counter
11 SABIRQ	Simultaneous PHASEA and PHASEB Change Interrupt Request Indicates that the PHASEA and PHASEB inputs changed simultaneously (within a single clock period). This event typically indicates an error condition, because quadrature coding requires only one of these inputs to change at a time. 0b - No simultaneous change has occurred 1b - A simultaneous change has occurred
10 SABIE	Simultaneous PHASEA and PHASEB Change Interrupt Enable Enables simultaneous PHASEA and PHASEB change interrupts when CTRL2[SABIRQ] becomes 1. 0b - Disable 1b - Enable
9 OUTCTL	Output Control Controls the pulsing of the POSMATCH output signal. This can control when a timer channel captures a timestamp. 0b - POSMATCH pulses when a match occurs between the position counters (POS) and the corresponding compare value (COMP) 1b - POSMATCH pulses when the UPOS, LPOS, REV, or POSD registers are read

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 REVMOD	Revolution Counter Modulus Enable Selects how Revolution Counter (REV) increments or decrements. By default, Revolution Counter (REV) is controlled based on the count direction and the INDEX pulse. As an option, you can control the revolution counter using the roll-over/under detection during modulo counting. 0b - Use INDEX pulse 1b - Use modulus counting roll-over or roll-under
7 ROIRQ	Roll-over Interrupt Request Indicates when the position counter (POS) rolls over from the MOD value to the INIT value or from FFFF_FFFFh to 0000_0000h. 0b - Did not occur 1b - Occurred
6 ROIE	Roll-over Interrupt Enable Enables roll-over interrupts when CTRL2[ROIRQ] is 1. This roll-over interrupt combines with the index interrupt signal. 0b - Disable 1b - Enable
5 RUIRQ	Roll-under Interrupt Request Indicates when the position counter (POS) rolls under from the INIT value to the MOD value or from 0000_0000h to FFFF_FFFFh. 0b - No roll-under has occurred 1b - Roll-under has occurred
4 RUIE	Roll-under Interrupt Enable Enables roll-under interrupts when CTRL2[RUIRQ] is 1. This roll-under interrupt combines with the index interrupt signal. 0b - Disable 1b - Enable
3 DIR	Count Direction Flag Indicates the direction of the last count. 0b - Down direction 1b - Up direction
2 MOD	Enable Modulo Counting Has multiple functions:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• When this field is 1, it allows Upper Position Counter (UPOS) and Lower Position Counter (LPOS) to count in a modulo fashion, using MOD and INIT as the upper and lower bounds of the counting range. During modulo counting: — When you indicate a "count up" and the position counter is equal to MOD, then the position counter reloads with the value of INIT. — When you indicate a "count down" and the position counter is equal to INIT, then the position counter reloads with the value of MOD. • When this field is 0, then the values of MOD and INIT are ignored, and the position counter wraps to zero when counting up from FFFF_FFFFh, or wraps to FFFF_FFFFh when counting down from 0. 0b - Disable 1b - Enable
1 UPDPOS	Update Position Registers Has multiple functions: • When this field is 1, it allows the TRIGGER input to write 0 to Position Difference Counter (POSD), Revolution Counter (REV), Upper Position Counter (UPOS), and Lower Position Counter (LPOS). • When this field is 0, Position Difference Counter (POSD), Revolution Counter (REV), Upper Position Counter (UPOS), and Lower Position Counter (LPOS) ignore the TRIGGER input. • If this field and CTRL2[INITPOS] are 1, Upper Position Counter (UPOS) and Lower Position Counter (LPOS) initialize to Upper Initialization (UINIT) and Lower Initialization (LINIT) on the rising edge of the TRIGGER input. Do not write 1 to this field when you use Last Edge Time (LASTEDGE) and Position Difference Period Counter (POSDPER) to measure speed. 0b - No action 1b - Clear
0 UPDHLD	Update Hold Registers Has multiple functions: • When this field is 1, it allows the TRIGGER input to update Position Difference Hold (POSDH), Revolution Hold (REVH), Upper Position Hold (UPOSH), and Lower Position Hold (LPOSH). • When this field is 0, the TRIGGER input does not update the hold registers (Position Difference Hold (POSDH), Revolution Hold (REVH), Upper Position Hold (UPOSH), and Lower Position Hold (LPOSH)). Do not write 1 to this field when you use Last Edge Time (LASTEDGE) and Position Difference Period Counter (POSDPER) to measure speed. Updating Position Difference Hold (POSDH) causes Position Difference Counter (POSD) to become 0. 0b - Disable 1b - Enable

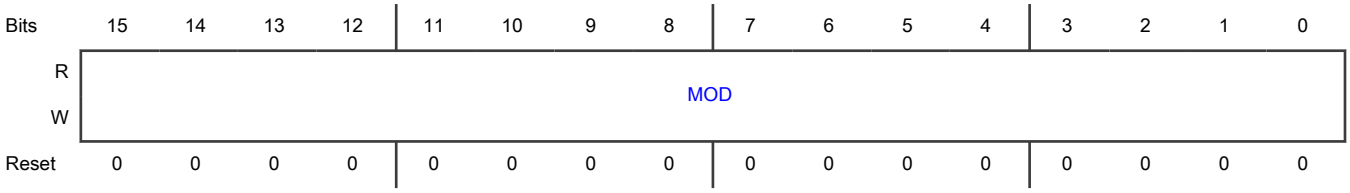
55.5.1.18 Upper Modulus (UMOD)

Offset

Register	Offset
UMOD	20h

Function
Contains the upper (most significant) half of the modulus register.

Diagram



Fields

Field	Function
15-0 MOD	MOD Specifies the upper (most significant) half of the modulus register. This field acts as the upper bound during modulo counting and as the upper reload value when rolling over from the lower bound.

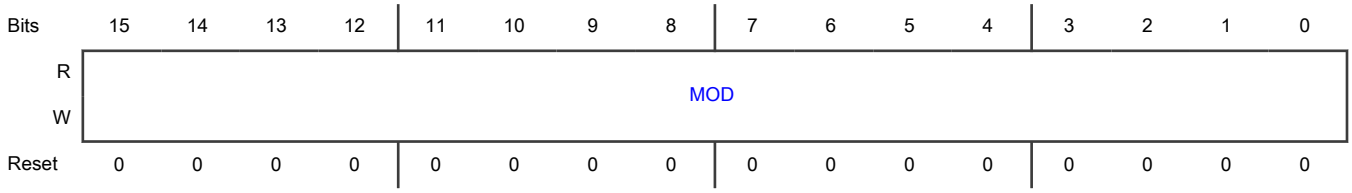
55.5.1.19 Lower Modulus (LMOD)

Offset

Register	Offset
LMOD	22h

Function
Contains the lower (least significant) half of the modulus register.

Diagram



Fields

Field	Function
15-0 MOD	MOD Specifies the lower (least significant) half of the modulus register. This field acts as the upper bound during modulo counting and as the upper reload value when rolling over from the lower bound.

55.5.1.20 Upper Position Compare (UCOMP)

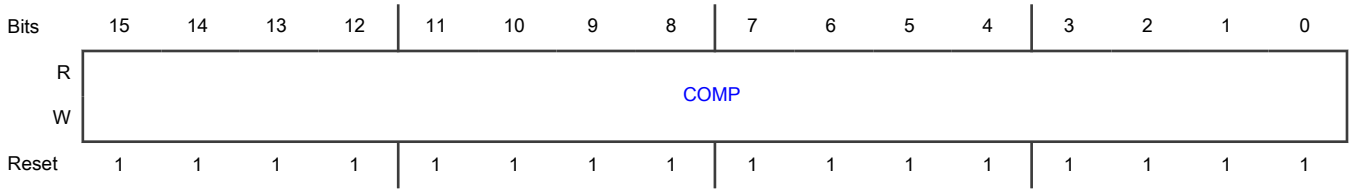
Offset

Register	Offset
UCOMP	24h

Function

Specifies the upper (most significant) half of the position compare register. When the value of the position counter (POS) matches the value of the position compare register (COMP), CTRL[CMPIRQ] becomes 1, and the POSMATCH output is asserted.

Diagram



Fields

Field	Function
15-0 COMP	COMP Specifies the upper (most significant) half of the position compare register.

55.5.1.21 Lower Position Compare (LCOMP)

Offset

Register	Offset
LCOMP	26h

Function

Contains the lower (least significant) half of the position compare register. When the value of the position counter (POS) matches the value of the position compare register (COMP), CTRL[CMPIRQ] becomes 1 and the POSMATCH output is asserted.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
15-0 COMP	COMP Specifies the lower (least significant) half of the position compare register.

55.5.1.22 Last Edge Time (LASTEDGE)

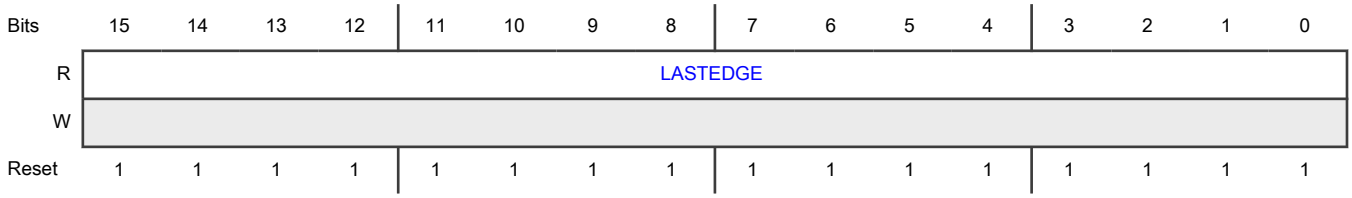
Offset

Register	Offset
LASTEDGE	28h

Function

Represents the time after the last edge occurred on PHASEA or PHASEB.

Diagram



Fields

Field	Function
15-0 LASTEDGE	Last Edge Time Counter Represents the time after the last edge occurred on PHASEA or PHASEB. This field counts up using the peripheral clock that CTRL3[PRSC] prescales. Any edge on PHASEA or PHASEB resets this field to 0 and starts counting. If the LASTEDGE count reaches FFFFh, the counting stops to prevent an overflow. Counting continues when an edge occurs on PHASEA or PHASEB.

55.5.1.23 Last Edge Time Hold (LASTEDGEH)

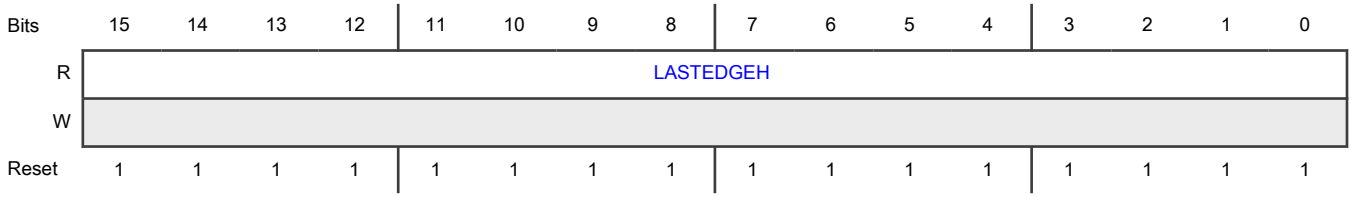
Offset

Register	Offset
LASTEDGEH	2Ah

Function

Holds register for the Last Edge Time (LASTEDGE) value.

Diagram



Fields

Field	Function
15-0 LASTEDGEH	Last Edge Time Hold Holds the Last Edge Time (LASTEDGE) value when you read Position Difference Counter (POSD).

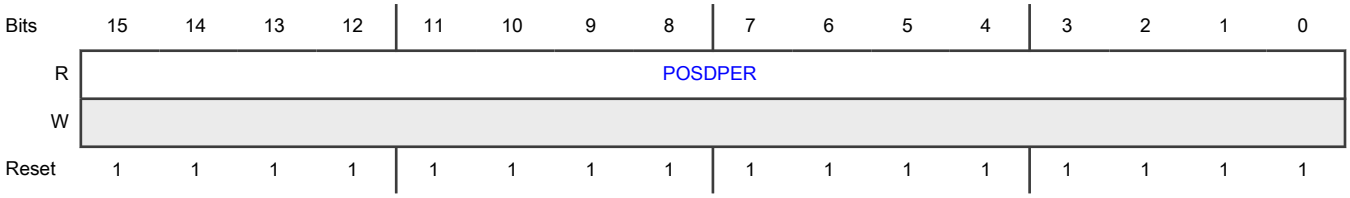
55.5.1.24 Position Difference Period Counter (POSDPER)

Offset

Register	Offset
POSDPER	2Ch

Function
Indicates the position-difference period.

Diagram



Fields

Field	Function
15-0 POSDPER	Position difference period Counts up using the peripheral clock that CTRL3[PRSC] prescales. Reading Position Difference Counter (POSD) loads the value of Last Edge Time (LASTEDGE) into Position Difference Period Counter (POSDPER). If Position Difference Period Counter (POSDPER) count reaches FFFFh, the counting stops to prevent an overflow. Counting continues when an edge occurs on PHASEA or PHASEB.

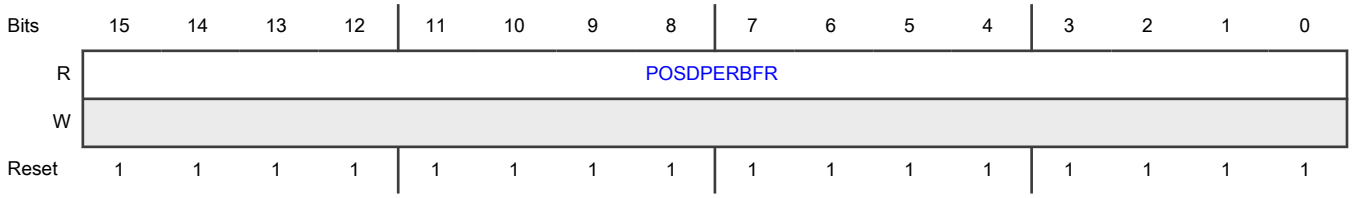
55.5.1.25 Position Difference Period Buffer (POSDPERBFR)

Offset

Register	Offset
POSDPERBFR	2Eh

Function
Buffers the value of Position Difference Period Counter (POSDPER).

Diagram



Fields

Field	Function
15-0 POSDPERBFR	Position difference period buffer Updates with Position Difference Period Counter (POSDPER) value when any edge occurs on PHASEA or PHASEB.

55.5.1.26 Position Difference Period Hold (POSDPERH)

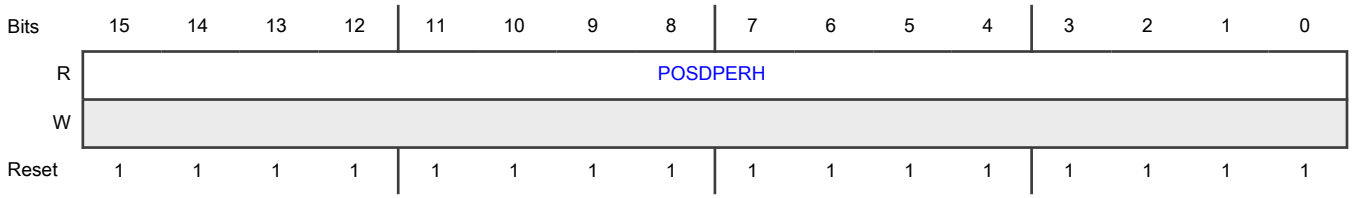
Offset

Register	Offset
POSDPERH	30h

Function

Holds register for Position Difference Period Counter (POSDPER) value.

Diagram



Fields

Field	Function
15-0 POSDPERH	Position difference period hold Updates with Position Difference Period Buffer (POSDPERBFR) value when you read Position Difference Counter (POSD).

55.5.1.27 Control 3 (CTRL3)

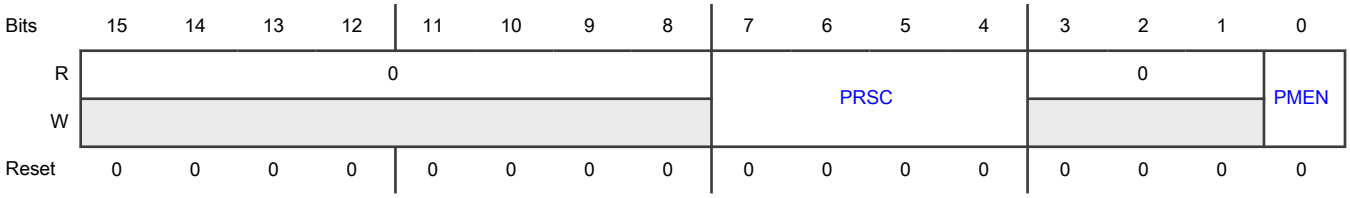
Offset

Register	Offset
CTRL3	32h

Function

Specifies prescaler values for the period clock divider and period measurement function.

Diagram



Fields

Field	Function
15-8 —	Reserved
7-4 PRSC	Prescaler Prescales the peripheral clock that LASTEDGE[LASTEDGE] and POSDPER[POSDPER] use. The resulting prescale clock value is 2^{PRSC}. The prescaler logic can divide the clock by a minimum of 1 and a maximum of 32,768.
3-1 —	Reserved
0 PMEN	Period Measurement Function Enable Specifies that the period-measurement functions of the following registers are used: • Last Edge Time (LASTEDGE) • Last Edge Time Hold (LASTEDGEH) • Position Difference Period Counter (POSDPER) • Position Difference Period Buffer (POSDPERBFR) • Position Difference Period Hold (POSDPERH) If this field is 0, Position Difference Counter (POSD) is loaded to Position Difference Hold (POSDH) and then becomes 0 whenever you read Position Difference Counter (POSD), Upper Position Counter (UPOS), Lower Position Counter (LPOS), or Revolution Counter (REV). If this field is 1, Position Difference Counter (POSD) is loaded to Position Difference Hold (POSDH) and then becomes 0 only when you read Position Difference Counter (POSD).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not used 1b - Used

Chapter 56
Real-Time Clock Subsystem (RTC_SUBSYSTEM)

56.1 Chip-specific RTC_SUBSYSTEM information

Table 453. Reference links to related information

Topic	Related module	Reference
Full description	RTC_SUBSYSTEM	RTC_SUBSYSTEM
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

56.1.1 Module instances

This device has one instance of the RTC_SUBSYSTEM, RTC_SUBSYSTEM0.

56.1.2 Power domain and clock

The RTC_SUBSYSTEM has three main components — the RTC, sub-second counter, and wake timer. The sub-second counter is implemented in the CORE_MAIN power domain, while the wake timer and critical portions of RTC are in the VBAT power domain.

The RTC_SUBSYSTEM receives FRO_16K and XTAL32K clock inputs. The RTC module's CTRL[CLK_SEL] register selects between the two clocks, and the selected clock is routed to the sub-second counter and wake timer.

56.2 Overview

RTC_SUBSYSTEM includes subsecond and wake timer features. The 16-bit subsecond up counter is clocked by the lp_osc input clock, which you must first enable (see Subsecond counter for more information). Either a 1 kHz or 0.5 kHz clock—which is the output of WAKE_TIMER_CTRL[OSC_DIV_ENA] (the clock divider of the lp_osc clock source)—clocks the wake timer.

56.2.1 Block diagram

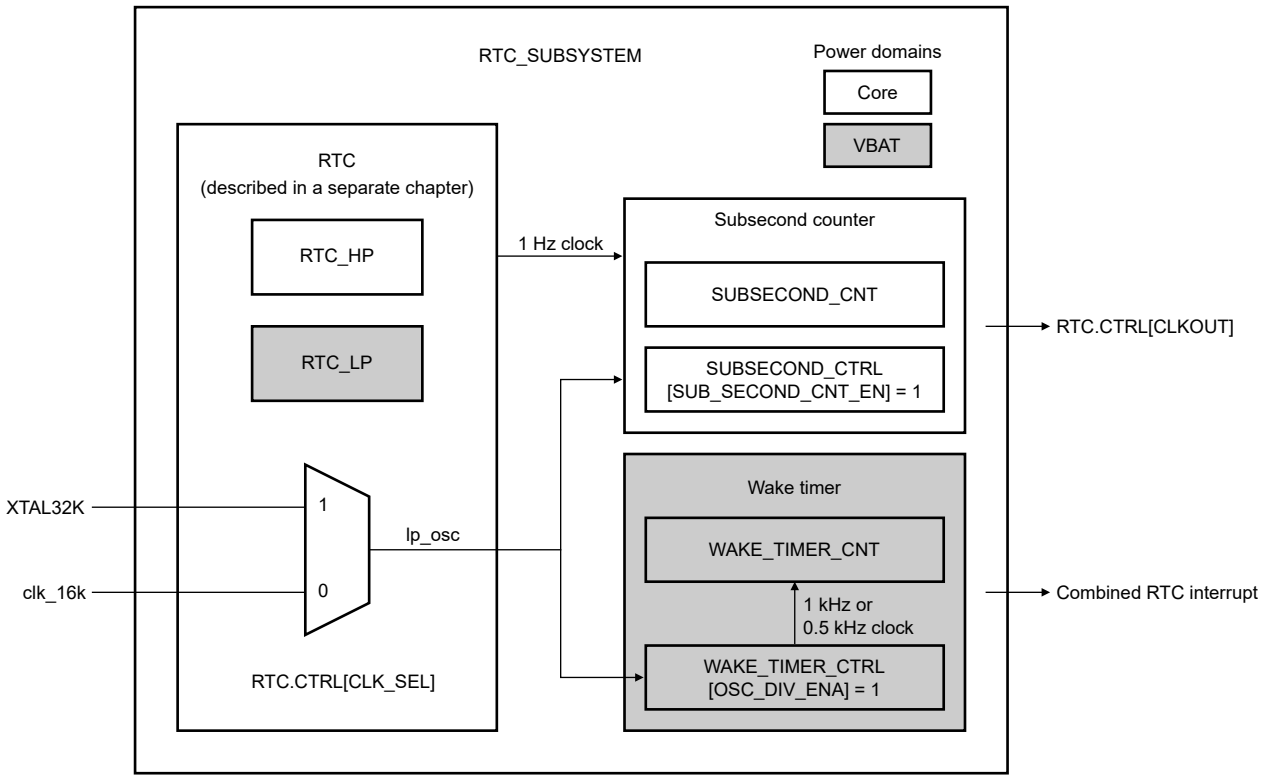


Figure 323. Block diagram

56.2.2 Features

- 16-bit subsecond counter function support in real-time application
- 32-bit wake-up timer counter in the VBAT domain to wake up the chip

56.3 Functional description

RTC_SUBSYSTEM includes a real-time clock (RTC), a subsecond counter, and a wake-up timer.

56.3.1 RTC

RTC handles the following:

- Basic clock functions
- Interrupts and alarms
- Time and calendaring functions and
- Clock compensation logic

See the "Real-Time Clock (RTC)" chapter for more information.

56.3.2 Subsecond counter

You must read the state of this counter through the bus and combine it with the 1-second RTC counter for a more precise time reading. The subsecond counter does not contribute to the alarm, interrupt, or wake-up generation.

SUBSECOND_CNT[SUBSECOND_CNT] is disabled when you reset RTC or disable the main RTC 1 Hz counter.
You must independently enable the field by writing 1 to SUBSECOND_CTRL[SUB_SECOND_CNT_EN] after enabling RTC.CTRL[CLKOUT] (see chip-specific RTC_SUBSYSTEM for more information) to select the 1 Hz clock output. After being enabled, the counter waits until the start of the next 1-second interval. Then, the counter begins incrementing at the lp_osc clock rate. As long as the counter remains enabled, SUBSECOND_CNT rolls over to 0 and resumes counting at the start of each 1-second interval.

56.3.3 Wake timer

The time interval required for many applications, including waking up the part from a low-power mode, often demands a greater degree of resolution than the 1-second minimum interval that the main RTC counter provides. For these applications, you can use a higher-frequency secondary timer that is an independent, standalone wake-up timer for intervals up to 64 s with approximately 1 ms of resolution.

Wake timer is a 16-bit down counter that clocks at 1 kHz after you enable it. If the frequency of the lp_osc clock source is 16 kHz, then the wake timer clocks at a rate of 0.5 kHz. Writing a nonzero value to this timer automatically enables the counter and launches a countdown sequence. When you use the counter as a wake timer, this write can occur prior to entering a low-power mode.

After a starting count value loads, the wake timer turns on, counts from the pre-loaded value down to 0, generates an interrupt and a wake command, and then turns itself off until a subsequent software write relaunches it.

56.3.4 Clocking

Table 454. Clocking

Clock	Description
lp_osc	Clock output from RTC.CTRL[CLOCK_SEL], which selects between the CLK_16K and XTAL32K inputs as the clock source.

56.4 External signals

This module has no external signals.

56.5 Initialization

This module does not require initialization.

56.6 RTC_SUBSYSTEM register descriptions

56.6.1 RTC_SUBSYSTEM memory map

RTC_SUBSYSTEM0 base address: 4004_C000h

Offset	Register	Width (In bits)	Access	Reset value
800h	Subsecond Control (SUBSECOND_CTRL)	32	RW	0000_0000h
804h	Subsecond Counter (SUBSECOND_CNT)	32	R	0000_0000h
C00h	Wake Timer Control (WAKE_TIMER_CTRL)	32	RW	0000_0000h
C0Ch	Wake Timer Counter (WAKE_TIMER_CNT)	32	RW	0000_0000h

56.6.2 Subsecond Control (SUBSECOND_CTRL)

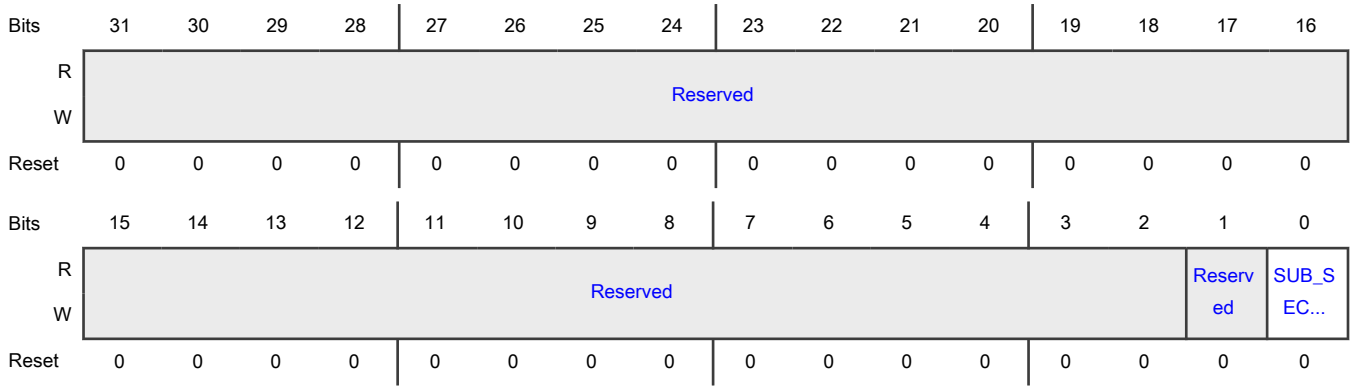
Offset

Register	Offset
SUBSECOND_CTRL	800h

Function

Controls Subsecond Counter (SUBSECOND_CNT).

Diagram



Fields

Field	Function
31-2 —	Reserved
1 —	Reserved
0 SUB_SECOND_CNT_EN	Subsecond Counter Enable **NOTE** Write 1 to RTC.CTRL[CLKOUT] to select the 1 Hz clock output for the subsecond counter to synchronize with the RTC_SECONDS counter. See the chip-specific RTC_SUBSYSTEM for more information. 0b - Disable 1b - Enable

56.6.3 Subsecond Counter (SUBSECOND_CNT)

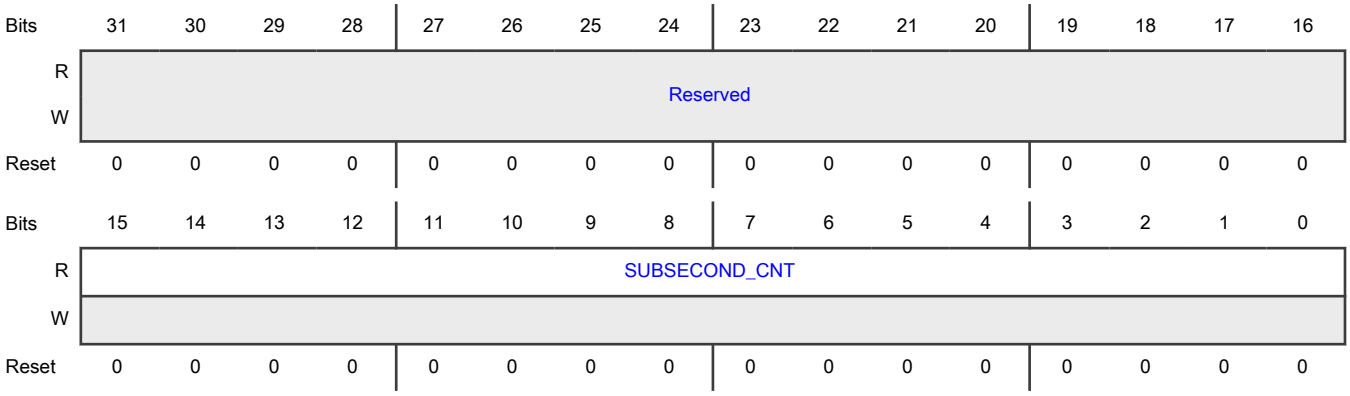
Offset

Register	Offset
SUBSECOND_CNT	804h

Function

Contains the subsecond counter value. The 32 kHz clock input clocks this 16-bit subsecond counter.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 SUBSECOND_CNT	Current Subsecond Counter Value Contains the current subsecond counter value. Because of clock domain crossing, you must read the register twice in succession and confirm whether the two values are the same. In case they are not, repeat the procedure until the values become the same.

56.6.4 Wake Timer Control (WAKE_TIMER_CTRL)

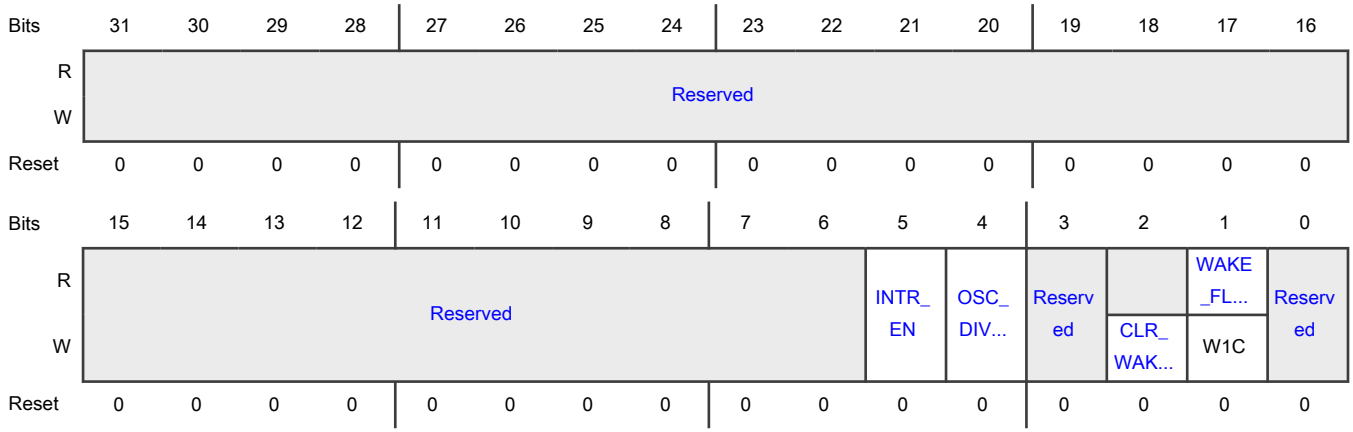
Offset

Register	Offset
WAKE_TIMER_CTRL	C00h

Function

Controls Wake Timer Counter (WAKE_TIMER_CNT). Either a 1 kHz or 0.5 kHz clock—which is the output of the 5-bit ripple counter with the lp_osc clock as the clock source—clocks the wake timer.

Diagram



Fields

Field	Function
31-6 —	Reserved
5 INTR_EN	Enable Interrupt Enables an interrupt when you write 1 to WAKE_TIMER_CTRL[WAKE_FLAG]. 0b - Disable 1b - Enable
4 OSC_DIV_ENA	OSC Divide Enable Enables the clock divider to divide down the lp_osc input clock source to generate either a 1 kHz or 0.5 kHz clock for the wake timer. Write 1 to OSC_DIV_ENA for the wake timer to run. 0b - Disable 1b - Enable
3 —	Reserved
2 CLR_WAKE_TIMER	Clear Wake Timer Clears the wake timer and halts the operation until a new count value is loaded. **NOTE** Reading this field always returns 0. 0b - No effect 1b - Clear the wake timer counter
1 WAKE_FLAG	Wake Timer Status Flag Specifies whether the wake timer has timed out.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not timed out 1b - Timed out
0 —	Reserved

56.6.5 Wake Timer Counter (WAKE_TIMER_CNT)

Offset

Register	Offset
WAKE_TIMER_CNT	C0Ch

Function

Contains the current value of the wake timer. Do not write to this register while counting is in progress.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WAKE_CNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WAKE_CNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 WAKE_CNT	Wake Counter Contains the current value of the wake timer. Because of clock domain crossing, you must read the register twice in succession and confirm whether the two values are the same. In case they are not, repeat the procedure until the values become the same. A write to this field preloads a start-count value into the timer and starts a countdown sequence.

Chapter 57
Real-Time Clock (RTC)

57.1 Chip-specific RTC information

Table 455. Reference links to related information

Topic	Related module	Reference
Full description	RTC	RTC
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

57.1.1 Module instances

This device has one instance of the RTC module, RTC0.

57.1.2 VBAT wakeup via WAKEUP_b pin

An RTC interrupt asserts the VBAT.STATUS[IRQ2_DET]. This interrupt can optionally be configured as a source to assert the WAKEUP_b pin. When the device is operating in the VBAT power mode, you can transition the device back to active mode by configuring an external circuitry to recognize the WAKEUP_b assertion and request for a system power on. See the VBAT chapter for more details.

57.2 Overview

Real Time Clock (RTC) is a low-power module that provides time keeping and calendaring functions, protection against spurious memory/register updates. It can also compensate the 1 Hz clock against variations in 32 kHz or 16 kHz clock in oscillator due to crystal or temperature.

57.2.1 Block diagram

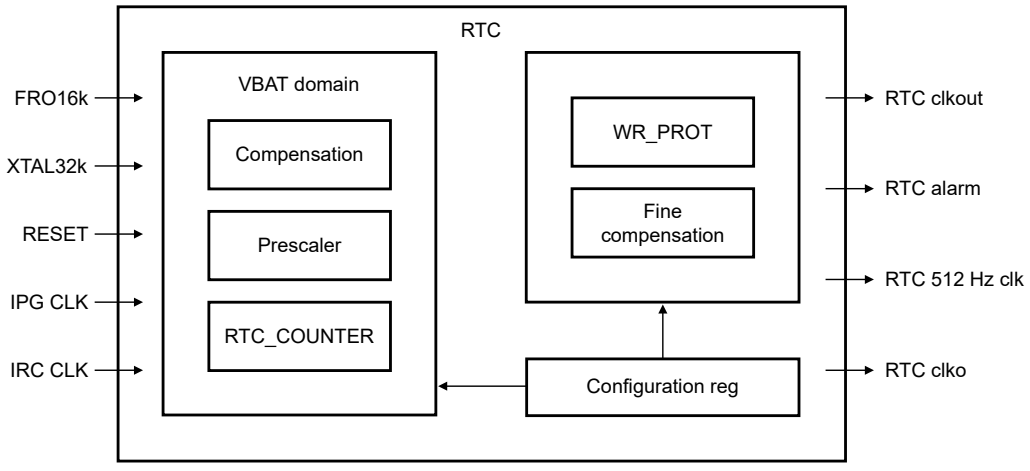


Figure 324. RTC block diagram

57.2.2 Features

RTC includes the following features:

- Designed for low power
  - Time and date counters are interleaved to prevent simultaneous toggling
- Basic clock functions
  - Separate counters for days, hour, minutes, and seconds
  - Calendaring support—Separate counters for year, month, and day of the week
    - Automatic adjustment for daylight saving with user-defined parameters
    - Automatic adjustment for month and leap year
  - External clock support to run the counters if you choose to provide an externally-compensated 1Hz clock
- Time zone offset—RTC uses local time which implicitly contains the time zone offset
- Programmable alarm with interrupt—Alarm is output from RTC in case the MCU uses it as a wake up event
- Periodic interrupts (sampling timer interrupts)
- Hardware compensation—Compensates 1 Hz clock (to the counters) against frequency variations in oscillator clock due to temperature changes or crystal characteristics. Programmable correction factor calculated by firmware.
- 16-bit CPU register programming interface with protection against runaway code
- Option to output the buffered 32.768 or 16.384 kHz clock or the compensated 1 Hz clock.

57.3 Functional description

57.3.1 Modes of Operation

This section describes the RTC modes of operation.

57.3.1.1 Wait mode/Power Down mode

RTC is fully operational in Wait mode (called Wait mode or Power Down mode depending on the device series). RTC runs off a 32.768 or 16.384 kHz clock that is not gated in Wait mode. This means the timekeeping functions (and other functions) that depend on the 32.768 or 16.384 kHz clock continue to operate independent of the CPU. Only the register block's clock is gated and no register contents are lost.

57.3.1.2 Stop mode/Deep Power Down mode

In Stop mode (called Stop mode or Deep Power Down mode depending on the device series), RTC is fully operational and behaves in the same way as it does when in Wait mode. If CPU power (VDD) is removed in Stop mode, RTC switches its power supply source to battery supply (VBAT) and isolates itself from the powered off domains.

NOTE

If the system bus clock is off in Stop mode, no read happens on RTC registers. When the system bus clock is switched on again, synchronization takes time. During the synchronization time, no write is allowed and any read to the RTC register returns FFFEh as the code to indicate the clock domains are synchronizing.

57.3.1.3 VBAT mode

Because it runs off a 32.768 or 16.384 kHz clock, RTC is fully operational in VBAT mode. The 32.768 or 16.384 kHz clock is not gated in VBAT mode and RTC's timekeeping functions (and other functions) that depend on the clock continue to operate independent of the CPU. Only the register block's clock is gated and no register contents are lost. After coming out of VBAT mode, the register content is not accessible for two osc_clk cycles due to synchronization.

57.3.2 Design overview

RTC provides basic timekeeping functions through counters for seconds, minutes, and hours, and calendaring functions via date, day-of-week, month, and year counters, including automatic adjustments for leap years and daylight saving time. Reading these counters indicates the current date and time and writing to these registers sets the date and time as provided by the user.

The alarm is set for a specific hour, minute, and second. If the alarm interrupt is enabled, an interrupt to CPU is generated when the time counters match the configured alarm hour, minute, and second settings. The alarm can also be configured to match a specific day, month, and year to generate an interrupt. The alarm signal is output to an MCU pin to allow wake up or control of external devices on board.

RTC also provides sampling timer interrupts.

A frequency compensation module is integrated into the RTC to correct any error in the 1 Hz clock due to variations in the 32.768/16.384 kHz clock which could be caused by crystal inaccuracy, board variations, or change in temperature. The combined compensation value for both crystal and temperature variation is determined by software and the correction is done in hardware.

The registers in RTC are configured via the CPU register programming interface. A protection mechanism is built into RTC to protect its registers against unintended writes by any runaway code. The protection mechanism requires the CPU to write a specific sequence of codes to the STATUS[7:6] bits to allow write access to the registers. After updating the registers, the CPU can write 10 to STATUS[7:6] bits to enable the write protection. After unlocking the registers, the CPU has a window of two seconds in which to update the register space. Following power-on reset, a window of 15 seconds is allowed for the CPU to configure the RTC, after which the registers are locked. Any updates beyond the unlock window require the CPU to unlock the registers, again.

For detailed description on the complete functionality of RTC, see Time and calendaring functions.

57.3.3 Clocking

The 32.768 kHz clock input from the oscillator is divided to generate a 512 Hz and a 1 Hz clock in RTC.

Table 456. Clocking

Signal name	Description	Direction
rtc_clk_out	1 Hz, 16 kHz, or 32.768 kHz clock output. Option for selecting compensated 1 Hz clock or buffered 32.768 kHz or 16 KHz clock is available in Control (CTRL).	Output
rtc_clko_s	Gated 32.768 kHz or 16 kHz clock output to other peripherals. Controlled by CTRL[CLKO_DIS].	Output
rtc_osc_32k_in_clk_hp	32.768 kHz clock input from oscillator, used in HP domain.	Input
rtc_osc_32k_in_clk_lp	32.768 kHz clock input from oscillator, used in LP domain.	Input
rtc_osc_16k_in_clk_hp	16.384 kHz clock input from FRO16, used in HP domain.	Input
rtc_osc_16k_in_clk_lp	16.384 kHz clock input from FRO16, used in LP domain.	Input
ipg_clk_hp_s	Gated system bus clock used for register read or write for high power or the CPU domain.	Input
ipg_clk_lp_s	Gated system bus clock used for register read or write for low power or the battery domain.	Input
ipg_clk_hp	System bus clock used to synchronize the oscillator 32.768 kHz or FRO 16 kHz clock to the ipg_clk domain (free running clock), used in HP domain.	Input
ipg_clk_lp	System bus clock used to synchronize the oscillator 32.768 kHz or FRO 16 kHz clock to the ipg_clk domain (free running clock), used in LP domain.	Input
rtc_irc_clk	Mhz clock used to fine compensate the oscillator clock.	Input

RTC has a 32.768 kHz clock input from the oscillator and a 16 kHz clock input from the FRO. You can select which clock to use by setting CTRL[CLK_SEL]. The clock becomes the output to the peripherals and divided to generate a 512 Hz clock and a 1 Hz clock. The 512 Hz clock becomes the output to peripherals and the 1 Hz clock is the input of the counter.

The 1 Hz clock is subject to variations in speed due to temperature changes or crystal irregularities. The input clock rtc_irc_clk runs in cycles of MHz and compensates for these variations by making precise adjustments to the 1 Hz clock.

RTC has one system bus clock used to synchronize the oscillator 32.768 kHz or 16 kHz clock to the ipg_clk domain.

Writing to CTRL[CLKOUT] configures either the clock input (32.768 kHz or 16 kHz) or the 1 Hz clock as the output from the SOC for use outside RTC. Writing to CTRL[CLKO_DIS] determines if the selected clock outputs to other peripherals.

57.3.4 Interrupts

Write one to IER[ALM_IE] to enable the alarm interrupt. The alarm interrupt is asserted when the programmed alarm value matches the counter values and the ISR[ALM_IS] bit is set. The status bit and interrupt are cleared by writing one. The status register is also cleared on software reset, except for the tamper status which remains unaffected. The counters matched for the alarm interrupt are selected based on the alarm type set in CTRL[ALM_MATCH].

There are eight periodic, sampling timer interrupts. When the clock generation counter counts to the programmed value, the corresponding bit in the status register is set. The status bits are cleared either by writing one or by software reset.

57.3.5 Time and calendaring functions

RTC performs and controls all the chronological functions listed below.

- Implements all counters for date and time and the related control logic
- Calculates leap year and adjusts the day count accordingly
- Increments or decrements counters to adjust for leap seconds
- Tracks the number of days in a month
- Automatically adjusts for daylight saving time
- Generates alarm with selectable matching of different counters.

Dynamic modifications to the date counters are done based on leap year, month, and daylight saving. Additionally, the user software can add or subtract a second to adjust for leap seconds. All changes are controlled by hardware and triggered by software.

A leap year is defined as a year in which the year value is divisible by 4 and 400 and has an extra day in February. Daylight saving is implemented in different regions of the world to shift the local time according to the summer or winter seasons. Time is shifted at a pre-defined time decided by the regional conditions and programmed by the user software. RTC automatically adjusts the time using hardware alarms for daylight saving. The day counter is also adjusted for months which have 28, 29, 30, or 31 days.

Alarm generation is done by the RTC block. Matching the counters is controlled by CTRL[ALM_MATCH] to give various alarm options for daily, monthly, yearly, or one-time alarms.

NOTE

Setting an hour alarm value that coincides with the daylight saving start hour value will not generate an alarm because the hour counter is incremented by two (instead of one) when daylight saving comes into effect. For example, setting ALM_HOURMIN to 0x1500 (hour alarm value = 15 or 3 PM) when DST_HOUR = 0x1411 (DST start hour value = 14 or 2 PM) will cause the hour counter to go from 14 to 16. The hour counter will not equal 15 and therefore the alarm which was set for 15 (or 3 PM) will not trigger. User software must take daylight saving changes into account when setting alarm values.

NOTE

User software must program an alarm time equal to the daylight saving end time (fallback time). The user software must write 0 to CTRL[DST_EN] when the alarm interrupt occurs, otherwise the correct operation might not take place.

57.3.6 RTC compensation logic

The compensation logic supports temperature and frequency compensation and provides an accurate and wide range of compensation suitable for many crystals. It can correct a wide range of crystal offsets from as low as 0.119 PPM.

There are two important components in the compensation logic: coarse compensation and fine compensation. The coarse compensation provides an accurate clock to RTC's internal time and date counters, and fine compensation generates an accurate 1 Hz clock output (via MCU pin) with high resolution clock edge placement (up to 0.88 ppm) and near 50 percent duty cycle.

Enabling the required compensation logic

Coarse compensation is enabled by writing to CTRL[COMP_EN] and fine compensation is enabled by writing to CTRL[FINEEN]. However, when FINEEN = 1, COMP_EN must be equal to 1. Setting both bits to 0 disables all compensation. This is the default state out of reset.

Compensation parameters need to be provided to allow required compensation logic to run. The parameters are provided by writing to the COMPEN register.

NOTE

If CTRL[FINEEN] = 0, then compensation must be disabled using CTRL[COMP_EN] before changing COMPEN for the first time. Successive changes can be done without disabling compensation.

If FINEEN = 1, then there are two options:

- Either clear previously accumulated fractional compensation value by disabling compensation from the CTRL register, program new values, and re-enable compensation and start a fresh
- Or, overwrite the current values leading to the addition of previously accumulated fractional compensation value to programmed fractional compensation value.

The integer part is always overwritten

Compensation parameters

Compensation logic requires the user software to provide compensation parameters in the COMPEN register in order to generate an accurate 1 Hz clock. These parameters are defined depending on the type of compensation enabled.

The compensation parameters (when FINEEN = 0) are defined as:

- **Compensation correction value:** Compensation correction value is a two's complement value by which the 1 Hz Clock is modified (during its generation) by either adding or removing RTC oscillator clock cycles.
- **Compensation interval:** Compensation interval is the duration in seconds over which the correction is applied. This is the time in which the addition or removal of 32.768 or 16.384 kHz clock cycles is done, thereby ensuring that the compensation interval is close to the interval obtained with an ideal 1 Hz clock.

The compensation parameters (when FINEEN = 1) are defined as:

- **Integral compensation value:** This is a two's complement value of the integer part of correction or compensation value that has to be adjusted in every 1 second period. This value is expressed in terms of the number of clock cycles of the RTC oscillator clock.
- **Fraction compensation value:** This is the fractional part of the correction or compensation value that has to be adjusted. This value is expressed as the number of clock cycles of a fixed 4.194304 MHz clock that have to be added. This value is always a positive number.

NOTE

When FINEEN = 1, the compensation interval is by default set to 1, indicating that the compensation will be done at every second.

Coarse compensation logic

Coarse compensation logic provides the accurate 1 Hz clock pulses to the time and date counters and the coarse 1 Hz clock to the fine compensation logic.

When FINEEN = 0: RTC compensates the clock over the provided compensation interval. The addition or removal of clocks is done in a single 1 Hz period, leaving the other 1 Hz periods of the compensation interval the same. This addition or removal of RTC oscillator clock cycles adjusts the 1 Hz clock in a way that keeps the overall compensation interval time close to the same interval being measured with an ideal clock.

When FINEN = 1: The integer part of the compensation (or correction) value is added or removed every 1 Hz period and the fraction part is accumulated and adjusted when the accumulated value equals 1 RTC oscillator clock cycle period.

To perform crystal offset compensation, the user software computes the compensation parameters external to RTC and, using details about the crystal characteristics (aging, drift, and so on), computes the correction factor. The user software then programs it in the two's complement format in the compensation register (COMPEN). Based on the values written in the COMPEN register, the compensation is performed.

To perform temperature compensation, the user software can maintain a lookup table in its memory which lists the change in frequency of 32.768 or 16.384 kHz crystal clock for each degree change in temperature. The CPU wakes up periodically to measure the external temperature via a temperature sensor connected to an A/D converter. The user software uses the lookup table to determine the compensation factor and writes the value to be compensated (in terms of number of 32.768 or 16.384 kHz clock cycles in two's complement format) in the compensation register (COMPEN). Based on the new values written, the compensation logic adjusts the clock from the next compensation interval.

The user software must compute a common compensation factor if it detects a variation in 32.768 or 16.384 kHz clock due to both temperature and crystal characteristics.

Vital statistics

- Range of compensation interval: 1 second to 255 seconds. If FINEEN = 0, writing a 0 disables compensation. If FINEEN = 1, this interval is always 1 second.
- Range of compensation: -128 to +127 (number of 32.768 or 16.384 kHz clock cycles)
- Selection criteria: Compensation is done only when enabled by user software. User software can disable compensation by writing a 0 compensation interval or setting CTRL[COMPEN] = 0.

Fine compensation logic

The fine compensation logic takes the coarse 1 Hz clock (from coarse compensation block) and generates accurate 1 Hz clock output with accurate clock edge placement and near 50 percent duty cycle.

The fine compensation runs when FINEEN = 1 and uses the MCU's IRC clock to adjust the fractional part of the compensation value every 1 Hz period. This provides an accurate edge placement on the 1 Hz clock.

The fine compensation module automatically adjusts the fractional compensation value for any variation in the IRC clock.

NOTE

Because the IRC clock is generated on MCU, this clock will be disabled when MCU power is OFF or in certain low power modes. In this case, the coarse 1 Hz clock is output from the MCU. The coarse 1 Hz clock is not a 50 percent duty cycle clock.

57.3.7 Write protection mechanism

The write protection mechanism protects the RTC registers from any unexpected updates that can happen from a runaway code. This logic monitors the values written to STATUS[WE]. By default, unconditional write access is allowed to these bits only. For writing the locking and unlocking sequence, 8-bit access should be configured in STATUS[7:0].

To enable write protection, write 10 to these bits. To disable write protection, write the following sequence to those bits: 00, 01, 11, 10.

After a power-on reset, the write protection mechanism is disabled, allowing the user software to configure the RTC block. After configuration is complete, the user code can enable the write protection mode. If this is not done by the user software, the registers are put into write protect mode 15 seconds after power-on. If the write protection mode is unlocked to update registers and not locked again by the user software, the write protection mode is automatically enabled after two seconds.

Any write access made to the registers when write protection is enabled (that is, registers are locked) will cause the transfer error signal to be asserted. Reads are allowed at all times.

NOTE

1. Always check STATUS[WRITE_PROT_EN] after running unlocking and locking sequences and re-run the sequence if STATUS[WRITE_ PROT_EN] is not in the required state.
2. Two consecutive unlocking sequences will lock the registers. After they are unlocked, running an unlock sequence before the timeout will lock RTC registers.

57.4 External Signals

Table 457. External signals

Signal	Description	Direction
RTC_CLKOUT	Output clock to external devices on the board.	Output

57.5 Memory map and registers

57.5.1 RTC register descriptions

The register address is the sum of the base address and the address offset. The base address is defined at the chip level. The address offset is defined at the module level.

NOTE

A 32.768 kHz or 16 kHz clock is needed to initialize the RTC. This clock can later be gated when programming the RTC registers.

NOTE

Use valid dates to configure registers. For example, do not configure a value like February 30, 2024, because there is no 30th in February 2024.

57.5.1.1 RTC memory map

RTC0 base address: 4004_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Year and Month Counters (YEARMON)	16	RW	0001h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
2h	Days and Day-of-Week Counters (DAYS)	16	RW	0001h
4h	Hours and Minutes Counters (HOURMIN)	16	RW	0000h
6h	Seconds Counters (SECONDS)	16	RW	0000h
8h	Year and Months Alarm (ALM_YEARMON)	16	RW	0000h
Ah	Days Alarm (ALM_DAYS)	16	RW	0000h
Ch	Hours and Minutes Alarm (ALM_HOURMIN)	16	RW	0000h
Eh	Seconds Alarm (ALM_SECONDS)	16	RW	0000h
10h	Control (CTRL)	16	RW	0000h
12h	Status (STATUS)	16	RW	0000h
14h	Interrupt Status (ISR)	16	RW	0000h
16h	Interrupt Enable (IER)	16	RW	0000h
1Ch	Sub Second Counter (RTC_TEST2)	16	R	0000h
22h	Daylight Saving Hour (DST_HOUR)	16	RW	0000h
24h	Daylight Saving Month (DST_MONTH)	16	RW	0000h
26h	Daylight Saving Day (DST_DAY)	16	RW	0000h
28h	Compensation (COMPEN)	16	RW	0000h

57.5.1.2 Year and Month Counters (YEARMON)

Offset

Register	Offset
YEARMON	0h

Function

Stores the value of the month and year counters. The year field stores the offset in years from a hardcoded base year (2112), not the actual year value. This is a signed value that ranges from -128 to +127. The actual year value can be calculated by adding the base year (2112) and the offset in the YEARMON[15:8] register, meaning the range of years supported is 1984 (2112 - 128) to 2239 (2112 + 127). For example, if the current year is 2007, then it will be represented in this register as 105 or 0x97. Software should program the offset from the base year into this register.

For year calculation:

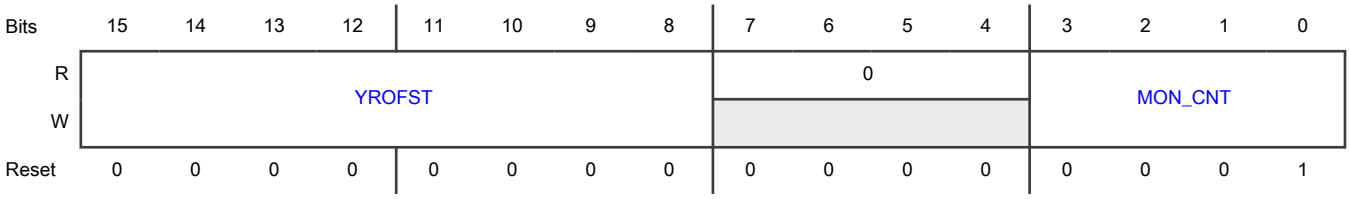
Actual Year = Base Year (for example, 2112) + Offset Year

The month field stores the count value of the months register. Writing to this register loads the months counter with this new value. The valid values are shown in table below.

User software should first determine the state of STATUS[INVAL_BIT] to determine if the counters are stable before a value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Both month and year are unaffected by software reset.

Diagram



Fields

Field	Function
15-8 YROFST	Year Offset Count Value Indicates the offset in years from the base year (hard coded as 2112) and does not show the actual year value. This is a signed value. Valid values are -128 to 127. The base year is 2112 and if the value of YEAR field is 0x10, the actual year is 2112 + 16 = 2128.
7-4 —	Reserved
3-0 MON_CNT	Month Counter Provides the value of the Months Counter. Valid Values are: 0000b,1101b,1110b,1111b - Illegal Value 0001b - January 0010b - February 0011b - March 0100b - April 0101b - May 0110b - June 0111b - July 1000b - August 1001b - September 1010b - October 1011b - November 1100b - December

57.5.1.3 Days and Day-of-Week Counters (DAYS)

Offset

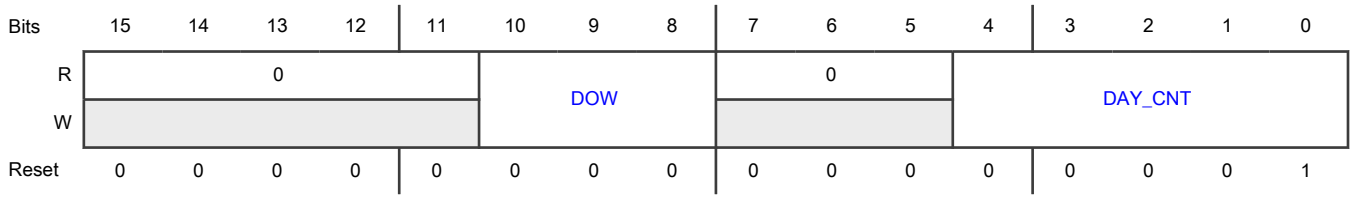
Register	Offset
DAYS	2h

Function

Shows the current value of the day-of-week counter and the days counter. Reading this register returns the latest value of the counters. Writing to this register loads the value to the day-of-week and days counters, and the counters continue to count from this new value. The day-of-week is not calculated automatically and should be written by CPU. This register is unaffected by software reset.

User software should first determine the state of STATUS[INVAL_BIT] to determine if the counters are stable before their value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Diagram



Fields

Field	Function
15-11 —	Reserved
10-8 DOW	Day of Week Counter Value 000b - Sunday 001b - Monday 010b - Tuesday 011b - Wednesday 100b - Thursday 101b - Friday 110b - Saturday 111b - Reserved
7-5 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
4-0 DAY_CNT	Days Counter Value Valid values are 1 to 31.

57.5.1.4 Hours and Minutes Counters (HOURMIN)

Offset

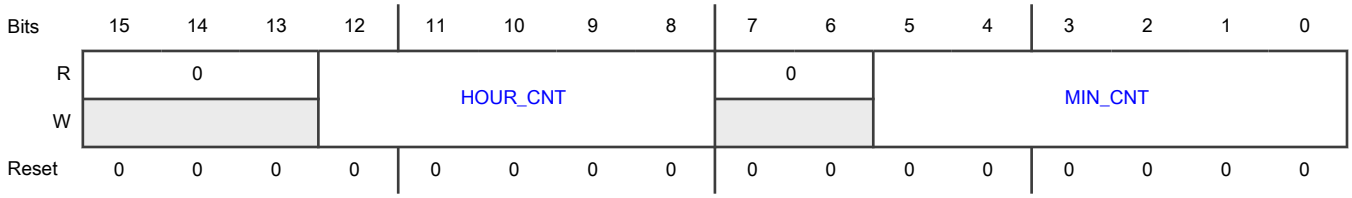
Register	Offset
HOURMIN	4h

Function

Used to program the hours and minutes counter. The register can be read anytime to get the current value of the counters. The hours counter can be set to anything between 0 and 23. The minutes counter can be set to anything between 0 and 59. This register is unaffected by software reset. Only power-on reset can reset this register.

User software should first determine the state of the STATUS[INVAL_BIT] to determine if the counters are stable before their value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Diagram



Fields

Field	Function
15-13 —	Reserved
12-8 HOUR_CNT	Hours Counter Value Valid count values are 0 to 23.
7-6 —	Reserved
5-0 MIN_CNT	Minutes Counter Value Valid count values are 0 to 59.

57.5.1.5 Seconds Counters (SECONDS)

Offset

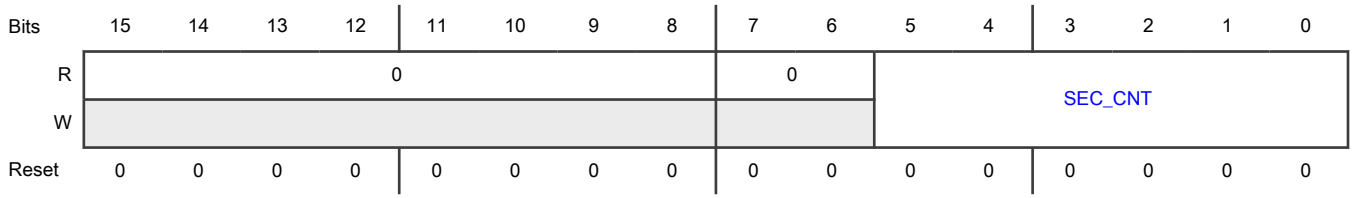
Register	Offset
SECONDS	6h

Function

Used to program the seconds counter. This register can be read anytime to get the current value of the counter. Seconds counter can be set to anything between 0 and 59 both included. This register is unaffected by software reset. Only power-on-reset can reset this register.

User software should first determine the state of the STATUS[INVAL_BIT] to determine if the counters are stable before their value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Diagram



Fields

Field	Function
15-8 —	Reserved
7-6 —	Reserved
5-0 SEC_CNT	Seconds Counter Value Valid count values are 0 to 59.

57.5.1.6 Year and Months Alarm (ALM_YEARMON)

Offset

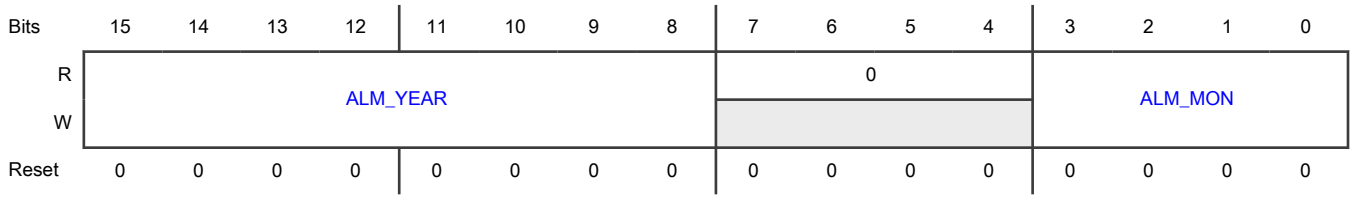
Register	Offset
ALM_YEARMON	8h

Function

Used to configure the months and year setting of the alarm. The alarm setting can be read or written anytime. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month, and year match their respective counter values. This register is reset to its default state on software reset.

User software can configure the alarm type by using the CTRL[ALM_MATCH].

Diagram



Fields

Field	Function
15-8 ALM_YEAR	Year Value for Alarm Same as years offset value in Year and Month Counters (YEARMON) .
7-4 —	Reserved
3-0 ALM_MON	Months Value for Alarm Same as months counter value in Year and Month Counters (YEARMON).

57.5.1.7 Days Alarm (ALM_DAYS)

Offset

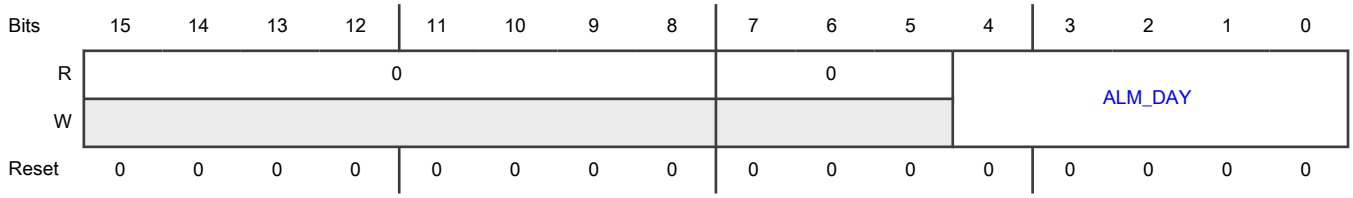
Register	Offset
ALM_DAYS	Ah

Function

Used to configure the day setting of the alarm. The alarm setting can be read or written anytime. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month, and year match their respective counter values. This register is reset to its default state on software reset.

User software can configure the alarm type using CTRL[ALM_MATCH].

Diagram



Fields

Field	Function
15-8 —	Reserved
7-5 —	Reserved
4-0 ALM_DAY	Days Value for Alarm Same as days counter value in Days and Day-of-Week Counters (DAYS) .

57.5.1.8 Hours and Minutes Alarm (ALM_HOURMIN)

Offset

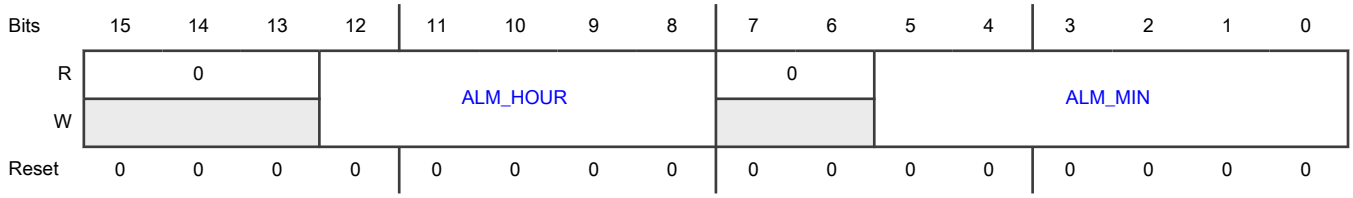
Register	Offset
ALM_HOURMIN	Ch

Function

Used to configure the hour and minute setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default state on software reset.

User software can configure alarm type using CTRL[ALM_MATCH].

Diagram



Fields

Field	Function
15-13	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
12-8 ALM_HOUR	Hours Value for Alarm Same as hours counter value in Hours and Minutes Counters (HOURMIN) .
7-6 —	Reserved
5-0 ALM_MIN	Minutes Value for Alarm Same as minutes counter value in Hours and Minutes Counters (HOURMIN) .

57.5.1.9 Seconds Alarm (ALM_SECONDS)

Offset

Register	Offset
ALM_SECONDS	Eh

Function

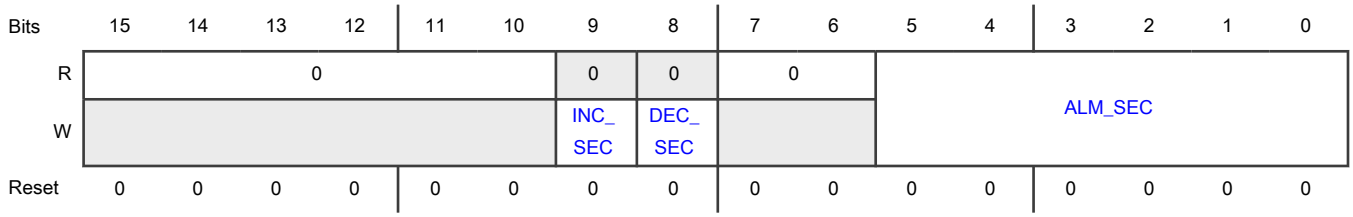
Used to configure the seconds setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default value on software reset.

User software has the option of using bits 9:8 to perform a correction on the seconds counter to compensate for leap seconds. Writing to these bits adds or subtracts 1 from the seconds counter and read returns zeros.

User software should first determine the state of STATUS[INVAL_BIT] to determine if the counters are stable before they can be incremented or decremented. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

User software can configure alarm type using CTRL[ALM_MATCH].

Diagram



Fields

Field	Function
15-10	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
9 INC_SEC	Increment Seconds Counter by 1. Controls the increment of seconds counter in case the software needs to make corrections to compensate for leap seconds or to perform fine trimming of time when needed. Writing to this bit increments the seconds counter and then the bit is cleared on the next posedge.
8 DEC_SEC	Decrement Seconds Counter by 1. Controls the decrement of the seconds counter in case software needs to make corrections for leap seconds or to perform fine trimming of time when needed. Writing to this bit has decrements the seconds counter and then the bit is cleared on the next posedge of the bus clock.
7-6 —	Reserved
5-0 ALM_SEC	Seconds Alarm Value Same as seconds counter value in Seconds Counters (SECONDS) .

57.5.1.10 Control (CTRL)

Offset

Register	Offset
CTRL	10h

Function

Governs all operations occurring inside the RTC. Used to specify the software reset, daylight controls, and the type of alarm function needed.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CLKOUT		0		CLKO_DIS	CLK_SEL	0	0	DST_EN	0	Reserved	ALM_MATCH		COMP_EN	FINEEN
W								SWR								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14-13 CLKOUT	RTC Clock Output Selection Selects which clock to output from SoC for use outside RTC. 00b - No output clock 01b - Fine 1 Hz clock with both precise edges 10b - 32.768 or 16.384 kHz clock 11b - Coarse 1 Hz clock with both precise edges
12-11 —	Reserved
10 CLKO_DIS	Clock Output Disable Determines whether the selected clock is output to other peripherals. 0b - The selected clock is output to other peripherals. 1b - The selected clock is not output to other peripherals.
9 CLK_SEL	RTC Clock Select Selects which clock is used by RTC 0b - 16.384 kHz clock is selected 1b - 32.768 kHz clock is selected
8 SWR	Software Reset Self clearing bit. Asserting this field clears the contents of alarm, interrupt (status and enable) registers, STATUS[CMP_DONE], and STATUS[BUS_ERR], and has no effect on DST, calendaring, time registers. 0b - Software Reset cleared 1b - Software Reset asserted
7 —	Reserved
6 DST_EN	Daylight Saving Enable Enables daylight saving function. The date and time for daylight saving changes are stored in the Daylight Saving Registers. These registers can be changed when this bit is 0. After this bit is set, those registers cannot be changed. When time and date match the values in those registers, daylight adjustment occurs. To disable daylight saving function, write 0 to this bit. 0b - Disabled. Daylight saving changes are not applied. Daylight saving registers can be modified. 1b - Enabled. Daylight saving changes are applied.
5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
4 —	Reserved
3-2 ALM_MATCH	Alarm Match Defines the type of alarm function. Selects which time and calendar counters are used for matching and will generate an alarm. 00b - Only seconds, minutes, and hours matched. 01b - Only seconds, minutes, hours, and days matched. 10b - Only seconds, minutes, hours, days, and months matched. 11b - Only seconds, minutes, hours, days, months, and year (offset) matched.
1 COMP_EN	Compensation Enable **NOTE** Hardware will not allow user software to write 1 to COMP_EN if both the FINEEN and COMP_EN bits are set to 1'b1. 0b - Coarse compensation is disabled. 1b - Coarse compensation is enabled.
0 FINEEN	Fine Compensation Enable 0b - Fine compensation is disabled 1b - Fine compensation is enabled.

57.5.1.11 Status (STATUS)

Offset

Register	Offset
STATUS	12h

Function

Indicates the status of various processes occurring inside the RTC. Also helps the user software to read time or date register when the values are stable and not changing. Compensation Done bit and Bus Error bit are cleared by writing 1. Software reset resets the whole register to its default state.

All memory mapped registers are protected against spurious updates by the write protect mechanism. To unlock the registers, a specific pattern (as mentioned in the table above) has to be written in the write enable bits (WE[1:0]) to enable or disable write protection. The WE[1:0] bits are the only bits that are freely writeable by user software. The write enable bits are self-clearing bits that always return zeros on read.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				CMP_DONE	0	0	BUS_ERR	0		CMP_INT	0	0		WRITE_P...	INVAL_B...
W					W1C			W1C	WE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-12 —	Reserved
11 CMP_DONE	Compensation Done Indicates that the current compensation cycle is complete. This field is cleared by writing 1. This bit is asserted seven 32.768 or 16.384 kHz clock cycles before the actual compensation interval completes so that back-to-back compensation can be enabled. 0b - Compensation busy or not enabled 1b - Compensation completed
10 —	Reserved
9 —	Reserved
8 BUS_ERR	Bus Error Indicates that a read or write cycle was initiated by software when the STATUS[INVAL_BIT] = 1. Write access to time or date registers is nullified (terminate normally) and no register value is changed. Reading when STATUS[INVAL_BIT] is asserted returns 0xFFFF. No transfer error is asserted. This bit is cleared by writing 1. 0b - Read and write accesses are normal. 1b - Read or write accesses occurred when STATUS[INVAL_BIT] was asserted.
7-6 WE	Write Enable Controls entry into and exit from the write protection mode. Both registers are protected by the write protection mechanism. These are self-clearing bits. Reads will return zeros. Disable write protection by writing this sequence: 00b, 01b, 11b, 10b. See Write protection mechanism for more information. **NOTE** When the registers are unlocked, they remain in this unlocked state for a time of 2 seconds, after which they are locked automatically. After power-on reset, the registers are unlocked and then are locked automatically after 15 seconds.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Enable Write Protection - Registers are locked.
5 CMP_INT	Compensation Interval Asserted for a time equal to compensation interval seconds as configured by the user. Used by the MCU to calculate the interrupts serviced during this interval and performs corrections in case of deviations (calibration). This bit toggles on at the start of every new compensation interval and is either 0 or 1 during the entire duration. This bit will not toggle if compensation logic has been disabled by MCU.
4 —	Reserved
3-2 —	Reserved
1 WRITE_PROT_EN	Write Protect Enable Status Indicates that registers are in locked mode and writing to them is disabled. Any write access made to the register space when write protection is enabled (that is, when they are in locked mode) will cause the transfer error signal to be asserted. 0b - Registers are unlocked and can be accessed. 1b - Registers are locked and in read-only mode.
0 INVAL_BIT	Invalidate CPU Read/Write Access Indicates the time and date counters are invalid or changing and therefore should not be read or written to. This bit is asserted for one oscillator clock cycle before and after the 1 Hz (seconds clock) boundary edge. Write access to time and date registers is nullified (terminate normally) and no register value is changed. Reading when STATUS[INVAL_BIT] is asserted returns 0xFFFF. No transfer error is asserted. 0b - Time and date counters can be read or written. Time and date is valid. 1b - Time and date counter values are changing or time and date is invalid and cannot be read or written.

57.5.1.12 Interrupt Status (ISR)

Offset

Register	Offset
ISR	14h

Function

NOTE

For the sampling timer interrupt status bits [14:6], see the chip configuration chapter for the applicable sampling timer frequencies.

Indicates the status of the various real-time clock interrupts. When an event of the types included in this register occurs, the bit will be set in this register regardless of the status of its corresponding interrupt enable bit. The status bits and interrupts are cleared by writing a value of 1. Interrupts may occur when the system clock is idle or in Standby mode. When the system enters the active power mode, an interrupt is indicated to the CPU. The first event of the Sampling Timer interrupts after power-on reset should not be used to qualify any periodic interval. However, the correct periodic interval (that is, 512 Hz or 256 Hz, and so on) should be determined using two sampling timer interrupts. The time between two interrupts should always be the correct time period.

This register is cleared on software reset.

NOTE

Sampling interrupts from 512 Hz to 2 Hz are generated using uncompensated clock. Only 1 Hz is compensated.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IS_512HZ	IS_256HZ	IS_128HZ	IS_64HZ	IS_32HZ	IS_16HZ	IS_8HZ	IS_4HZ	IS_2HZ	IS_1HZ	MIN_IS	HOUR_IS	DAY_IS	ALM_IS	0	0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15 IS_512HZ	512 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
14 IS_256HZ	256 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
13 IS_128HZ	128 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
12 IS_64HZ	64 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
11 IS_32HZ	32 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
10 IS_16HZ	16 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 IS_8HZ	8 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
8 IS_4HZ	4 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
7 IS_2HZ	2 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
6 IS_1HZ	1 Hz Interval Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
5 MIN_IS	Minutes Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
4 HOUR_IS	Hours Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
3 DAY_IS	Days Interrupt Status 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
2 ALM_IS	Alarm Interrupt Status Indicates that the alarm value programmed matches the counter values. 0b - Interrupt is de-asserted. 1b - Interrupt is asserted.
1 —	Reserved
0 —	Reserved

57.5.1.13 Interrupt Enable (IER)

Offset

Register	Offset
IER	16h

Function

NOTE

For the sampling timer interrupt enable bits [14:6], see the chip configuration chapter for the applicable sampling timer frequencies.

Enables and disables the various real-time clock interrupts. De-asserting an interrupt enable bit has no effect on the assertion of its corresponding status bit.

Alarm interrupt is asserted on counters matching the alarm setting done in the memory map. The counters matched for the alarm interrupt are selected based on the alarm type set in CTRL[3:2]. The various types of alarms available are shown in the following table. Only one alarm type can be used at a time.

Table 458. Alarm Match Table

ALM_MATCH[1:0] (CTRL[3:2])	Counters Matched	Alarm Type
00	Seconds, Minutes, and Hours	Daily
01	Seconds, Minutes, Hours, and Days	Monthly
10	Seconds, Minutes, Hours, Days, and Months	Yearly
11	Seconds, Minutes, Hours, Days, Months, and Year	One Time

RTC generates a common interrupt. The user software should read the status register in the interrupt service routine to determine which interrupt has occurred.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IE_512HZ	IE_256HZ	IE_128HZ	IE_64HZ	IE_32HZ	IE_16HZ	IE_8HZ	IE_4HZ	IE_2HZ	IE_1HZ	MIN_IE	HOUR_IE	DAY_IE	ALM_IE	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15 IE_512HZ	512 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
14 IE_256HZ	256 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
13 IE_128HZ	128 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
12 IE_64HZ	64 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
11 IE_32HZ	32 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
10 IE_16HZ	16 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
9 IE_8HZ	8 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
8 IE_4HZ	4 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
7 IE_2HZ	2 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
6 IE_1HZ	1 Hz Interval Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
5 MIN_IE	Minutes Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
4 HOUR_IE	Hours Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Interrupt is disabled. 1b - Interrupt is enabled.
3 DAY_IE	Days Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
2 ALM_IE	Alarm Interrupt Enable Indicates that the alarm value programmed matches the counter values. 0b - Interrupt is disabled. 1b - Interrupt is enabled.
1 —	Reserved
0 —	Reserved

57.5.1.14 Sub Second Counter (RTC_TEST2)

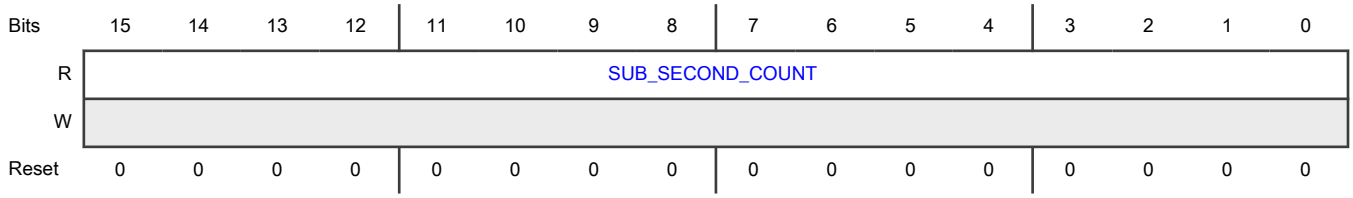
Offset

Register	Offset
RTC_TEST2	1Ch

Function

The RTC_TEST2 counter is used as a subsecond counter.

Diagram



Fields

Field	Function
15-0	Sub Second Counter Value

Table continues on the next page...

Field	Function
SUB_SECOND_COUNT	This counter is used as a sub second counter. Actually it counts the 32 kHZ/16 kHZ clk cycles and auto clear every second.

57.5.1.15 Daylight Saving Hour (DST_HOUR)

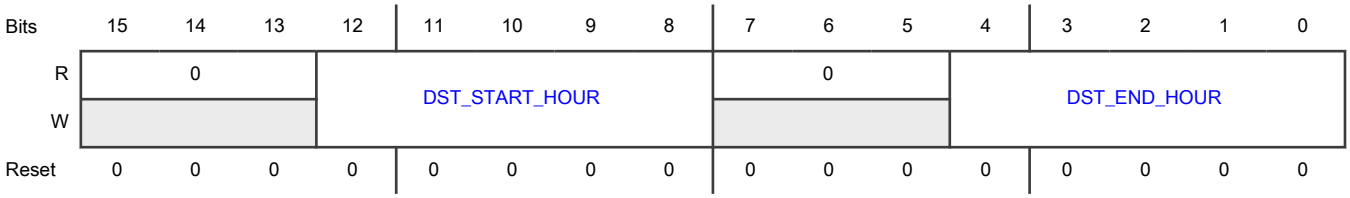
Offset

Register	Offset
DST_HOUR	22h

Function

Stores the time in hours when daylight saving has to be applied or reversed. This register is programmable when CTRL[DST_EN] = 0. When CTRL[DST_EN] = 1, the contents of this register cannot be changed. The user software should program the correct hour value (0-23) per the regional settings. For example, if daylight saving starts at 2:00 AM on March 25 and ends at 2:00 AM on October 28 in 2007 then the time at which the RTC advances or falls back is actually 1:59 AM. Therefore, the user software should program one, not two, for the hour count value (that is, write 0x0101) in this register. A 59-minute count is automatically checked inside RTC and not required to be programmed. This register has no effect on software reset.

Diagram



Fields

Field	Function
15-13 —	Reserved
12-8 DST_START_HOUR	Daylight Saving Time (DST) Hours Start Value The hour value for the time when DST comes into effect. Same as hours counter value in the Hours and Minutes Counters (HOURMIN).
7-5 —	Reserved
4-0 DST_END_HOUR	Daylight Saving Time (DST) Hours End Value The hour value for the time when DST is reversed. Same as hours counter value in the Hours and Minutes Counters (HOURMIN).

57.5.1.16 Daylight Saving Month (DST_MONTH)

Offset

Register	Offset
DST_MONTH	24h

Function

Stores the month when the daylight saving needs to be applied or reversed. This register is programmable when CTRL[DST_EN] = 0. When CTRL[DST_EN] = 1, the contents of this register cannot be changed. The CPU should program the correct month value (1-12) per the regional settings. For example, if daylight saving starts on March 25 and ends on October 28 in 2007, then CPU should write 0x030A in this register. This register has no effect on software reset.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				DST_START_MONTH				0				DST_END_MONTH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-12 —	Reserved
11-8 DST_START_MONTH	Daylight Saving Time (DST) Month Start Value The month value for the time when DST comes into effect. See Year and Month Counters (YEARMON).
7-4 —	Reserved
3-0 DST_END_MONTH	Daylight Saving Time (DST) Month End Value The month value for the time when DST is reversed. See Year and Month Counters (YEARMON).

57.5.1.17 Daylight Saving Day (DST_DAY)

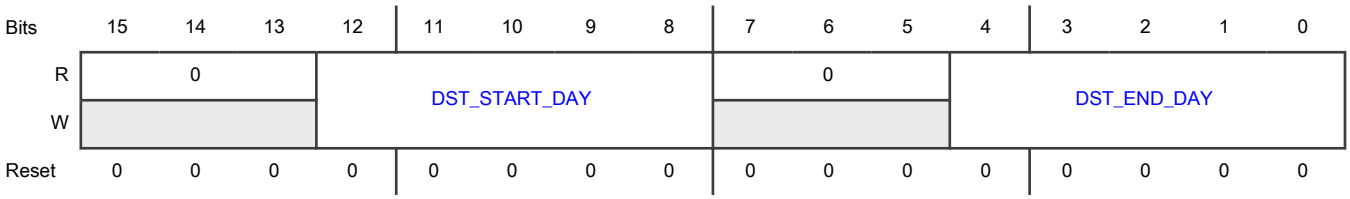
Offset

Register	Offset
DST_DAY	26h

Function

Stores the day when the daylight saving needs to be applied or reversed. This register is programmable when CTRL[DST_EN] = 0. When CTRL[DST_EN] = 1, the contents of this register cannot be changed. The CPU should program the correct day value (1-31) per the regional settings. For example, if the Daylight Saving starts at March 25 and ends at October 28 in 2007, then the CPU should write 0x191C in this register. This register is unaffected by software reset.

Diagram



Fields

Field	Function
15-13 —	Reserved
12-8 DST_START_DAY	Daylight Saving Time (DST) Day Start Value The day value for the time when DST comes into effect.
7-5 —	Reserved
4-0 DST_END_DAY	Daylight Saving Time (DST) Day End Value The day value for the time when DST is reversed.

57.5.1.18 Compensation (COMPEN)

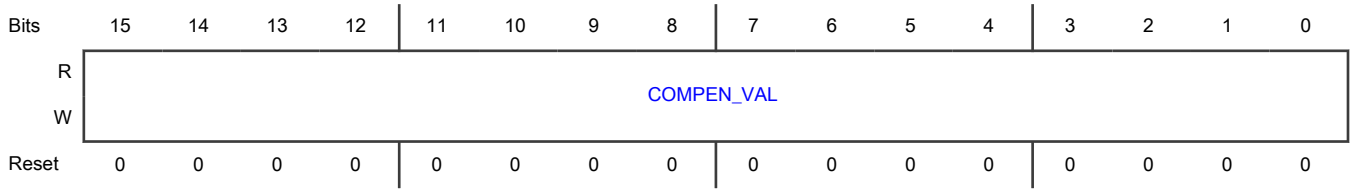
Offset

Register	Offset
COMPEN	28h

Function

Stores the compensation value to be used by the compensation block to correct the 1 Hz clock.

Diagram



Fields

Field	Function
15-0 COMPEN_VAL	Compensation Value Stores the compensation parameters. The definition of this field is dependent on the setting of CTRL[FINEEN]. If CTRL[FINEEN] = 0: • Compensation or Correction Value (COMPEN[7:0]) — Compensation or correction value is a two's complement value that modifies the 1 Hz clock (during its generation) by either adding or removing RTC oscillator clock cycles. • Compensation Interval (COMPEN[15:8]) — Compensation interval is the duration in seconds over which the correction is applied. This is the time in which the addition or removal of 32.768 or 16.384 kHz clock cycles occurs, thereby ensuring that the compensation interval is close to the interval obtained with an ideal 1 Hz clock. If CTRL[FINEEN] = 1 • Integral Compensation Value (COMPEN[15:12]) — This is a two's complement value of the integer part of correction or compensation value that has to be adjusted in every 1 second period. This value is expressed in terms of number of clock cycles of the RTC oscillator clock. • COMPEN[11:7] — Should be zero • Fraction Compensation Value (COMPEN[6:0]) — This is the fractional part of the correction or compensation value that has to be adjusted. This value is expressed as number of clock cycles of a fixed IRC_CLK. This value is always a positive number.

Chapter 58
Frequency Measurement (FREQME)

58.1 Chip-specific Frequency Measurement information

Table 459. Reference links to related information

Topic	Related module	Reference
Full description	FREQME	Frequency Measurement
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See FREQMEAS_REF and FREQMEAS_TAR registers in INPUTMUX for a list of clocks that can be selected as inputs to the frequency measurement circuit.

58.1.1 Module instances

This device has one instance of the Frequency Measurement module, FREQME0.

58.2 Overview

FREQME accurately measures the frequency of an on- or off-chip target clock signal using a selectable on-chip reference clock. For example, it can accurately determine the frequency of a low-power oscillator that varies depending on process and temperature.

58.2.1 Block diagram

Figure 325. Block diagram

See the chip-specific FREQME information for the target and reference clock selection options.

58.2.2 Features

- High-accuracy Frequency Measurement mode for on- and off-chip clocks
- Pulse Width Measurement (PWM) mode
- Reference and target clock inputs selectable from among various chip-specific options
- Optional measurement complete interrupt
- Result out-of-range detection with optional interrupt

58.3 Functional description

The following sections describe FREQME functional details.

58.3.1 Frequency Measurement mode

In Frequency Measurement mode (CTL_W[PULSE_MODE] = 0), FREQME counts the number of target clock cycles that occur during a specified number of cycles from a reference clock that has a known frequency. You can then calculate the target clock frequency based on the frequency of the reference clock, the number of reference clock cycles, and the number of target clock cycles (see Equation 26).

The frequency measurement circuit is based on two 31-bit counters—one clocked by the selected reference clock and one by the selected target clock. You can only read the target clock counter. FREQME synchronizes the clocks at the start and end of each count sequence during preparation for the next count sequence. A count sequence consists of incrementing the target clock counter for each target clock pulse that occurs during the time defined by $2^{CTRL_W[REF_SCALE]}$ periods of the reference clock.

After selecting reference and target clocks, you initiate a measurement cycle by writing 1 to CTRL_W[MEASURE_IN_PROGRESS]. You can then poll this same field, which automatically transitions to 0 when the measurement operation completes. Alternatively, the completion of the measurement operation can generate an interrupt.

The measurement cycle terminates when the reference counter equals the value $2^{CTRLSTAT[REFSCALE]}$. If CTRLSTAT[REFSCALE] = 0, the reference counter counts to one and stops. You can use this feature to measure the frequency of a fast target clock using a slow reference clock such as the 32 kHz clock without taking much time for the measurement to complete. The penalty is reduced accuracy in the measurement.

When the counting operation completes, the state of the target counter is loaded into CTRL_R[RESULT], and the CTRL_W[MEASURE_IN_PROGRESS] field is 0. You can then read the value and calculate the target frequency as follows, with the frequencies given in MHz.

$$F_{target} = (CTRL_R[RESULT] - 2) \times F_{reference} \div 2^{CTRLSTAT[REFSCALE]}$$

Equation 26. Calculating the target clock frequency

58.3.1.1 Accuracy

The Frequency Measurement mode can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) with a high degree of accuracy by using an on-chip clock of known frequency as a reference clock.

Uncertainty in the reference clock (for example a ±1 % accuracy of the clock) adds to the measurement error of the target clock. In general, though, this additional error is less than the uncertainty of the reference clock.

58.3.2 Pulse Measurement mode

When you write 1 to CTRL_W[PULSE_MODE] and write 1 to CTRL_W[MEASURE_IN_PROGRESS] to start a measurement cycle, FREQME counts target clock pulses while the reference clock is in a specific state (high or low), selected by writing to CTRL_W[PULSE_POL]. See the description of these fields for more details.

58.3.3 Clocking

The clock inputs are as follows:

- Reference clock (REF_CLK_IN). This clock has a known frequency.
- Target clock (TARGET_CLK_IN). This clock frequency can be calculated based on the reference clock.

NOTE

These clocks can be all synchronous.

See the chip-specific FREQME information for the target and reference clock selection options.

58.3.4 Interrupts

The following interrupts can optionally be generated at the completion of a frequency measurement cycle.

Table 460. Interrupts

Condition	Interrupt enable field	Interrupt status field
Result is ready to read	CTRL_W[RESULT_READY_INT_EN]	CTRLSTAT[RESULT_READY_INT_EN]
Result is greater than MAX[MAX_VALUE]	CTRL_W[GT_MAX_INT_EN]	CTRLSTAT[GT_MAX_INT_EN]
Result is less than MIN[MIN_VALUE]	CTRL_W[LT_MIN_INT_EN]	CTRLSTAT[LT_MIN_INT_EN]

58.4 External signals

Table 461. External signals

Signal	Description	Direction
REF_CLK_IN	Reference clock. On-chip clock with known frequency chosen as reference.	Input
TARGET_CLK_IN	Target clock. On- or off-chip clock to be measured by FREQME.	Input

58.5 Initialization

Initialize FREQME by performing the following steps:

1. Enable the clock that drives FREQME in the chip-level clock control register. This step enables the register interface and the peripheral function clock.
2. Clear the FREQME peripheral reset in the chip-level reset control register.
3. If measuring an external clock, use the Input/Output pin configuration registers to connect the FREQME target clock input (TARGET_CLK_IN) to an external pin.
4. Ensure that the reference and target clocks are enabled.
5. Select the reference and target clocks.

58.6 Memory map and register definition

This section includes the FREQME memory map and detailed descriptions of all registers.

58.6.1 FREQME register descriptions

58.6.1.1 FREQME memory map

FREQME0 base address: 4001_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (in Read mode) (CTRL_R)	32	R	0000_0000h
0h	Control (in Write mode) (CTRL_W)	32	W	0000_0000h
4h	Control Status (CTRLSTAT)	32	RW	0000_0000h
8h	Minimum (MIN)	32	RW	0000_0000h
Ch	Maximum (MAX)	32	RW	7FFF_FFFFh

58.6.1.2 Control (in Read mode) (CTRL_R)

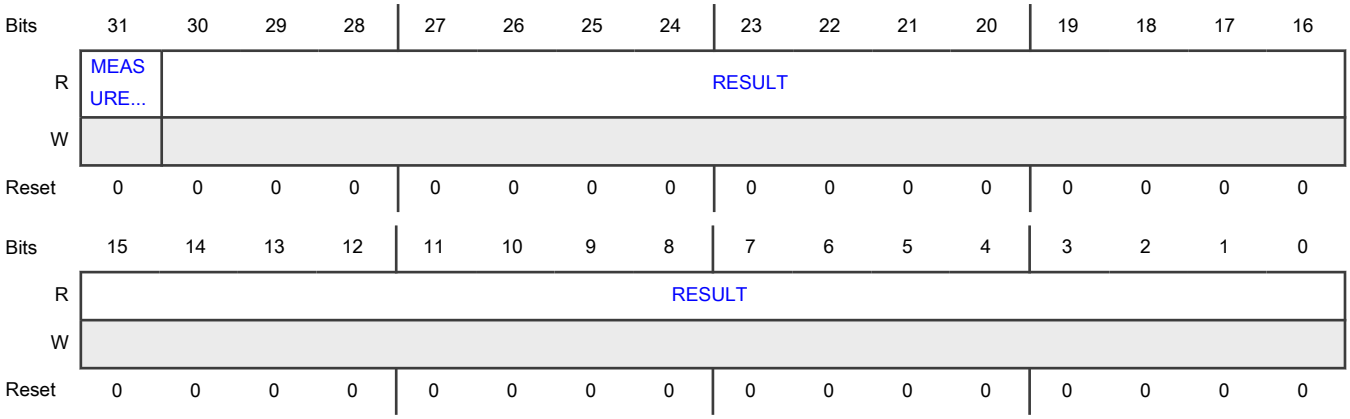
Offset

Register	Offset
CTRL_R	0h

Function

Contains different fields depending on whether you are performing a read or a write. Reading this register provides the measurement results and the status of the measurement cycle.

Diagram



Fields

Field	Function
31 MEASURE_IN_PROGRESS	Measurement In Progress Indicates whether measurement is in progress or complete. If complete, you can read the result from RESULT. 0b - Complete 1b - In progress
30-0 RESULT	Indicates the measurement result—either the target clock counter value (for Frequency Measurement mode) or pulse width measurement (for Pulse Width Measurement mode). This field is valid only when MEASURE_IN_PROGRESS = 0, In Continuous mode (CTRL_W[CONTINUOUS_MODE_EN] = 1), the value is the result from the most-recently completed measurement.

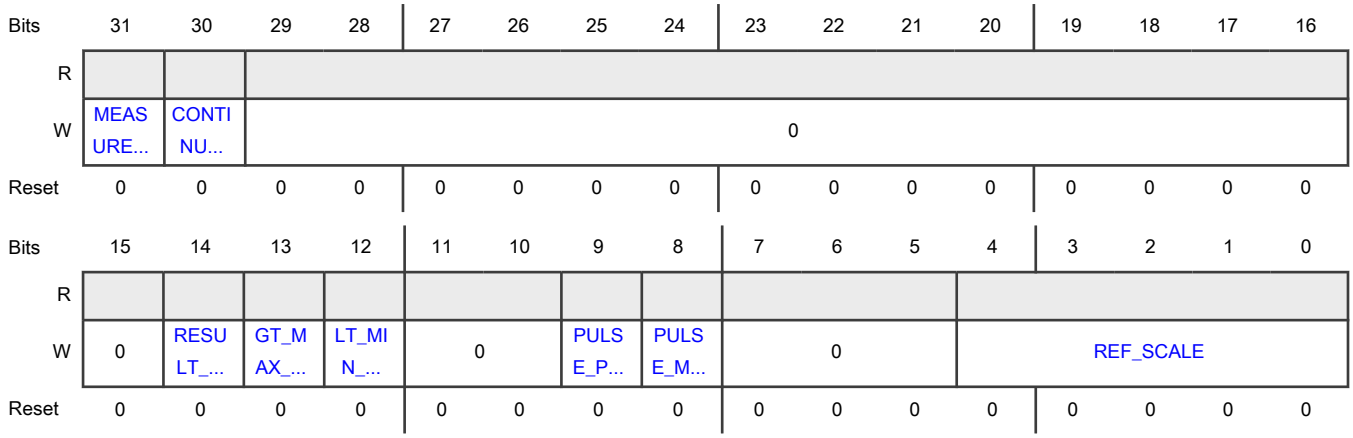
58.6.1.3 Control (in Write mode) (CTRL_W)

Offset

Register	Offset
CTRL_W	0h

Function
Writes to this register control and configure the measurement cycle.

Diagram



Fields

Field	Function
31 MEASURE_IN_PROGRESS	Measurement In Progress Initiates a frequency measurement cycle or terminates a measurement cycle that is in progress. Writing 1 to this field initiates a frequency or pulse width measurement process. Hardware automatically writes 0 to the MEASURE_IN_PROGRESS field when the measurement cycle completes. If there is an active measurement in progress, a new measurement starts. Writing 0 to this field forces the termination of any measurement cycle currently in progress and resets CTRL_R[RESULT] or just resets CTRL_R[RESULT] if idle. 0b - Terminates measurement 1b - Initiates measurement
30 CONTINUOUS_MODE_EN	Continuous Mode Enable When you write 1 to both MEASURE_IN_PROGRESS and this field, measurement is performed continuously. The result for the most-recently completed measurement is available in CTRL_R[RESULT]. When the result is out of range (that is, CTRLSTAT[GT_MAX_STAT] = 1 or CTRLSTAT[LT_MIN_STAT] = 1), this field automatically transitions to 0 and the continuous measurement is suspended. After clearing both status bits, you must write 1 to both MEASURE_IN_PROGRESS and this field restart measurement in continuous mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
29-15 —	Reserved
14 RESULT_READY_INT_EN	Result Ready Interrupt Enable Generates interrupt when a measurement completes and the result is ready (CTRLSTAT[RESULT_READY_STAT] = 1). 0b - Disable 1b - Enable
13 GT_MAX_INT_EN	Greater Than Maximum Interrupt Enable Generates an interrupt when the result is greater than MAX[MAX_VALUE] (CTRLSTAT[GT_MAX_STAT] = 1). 0b - Disable 1b - Enable
12 LT_MIN_INT_EN	Less Than Minimum Interrupt Enable Generates an interrupt when the result is less than MIN[MIN_VALUE] (CTRLSTAT[LT_MIN_STAT] = 1). 0b - Disable 1b - Enable
11-10 —	Reserved
9 PULSE_POL	Pulse Polarity Specifies whether a pulse width (high period or low period) of the reference clock is measured in the Pulse Width Measurement mode. **NOTE** PULSE_POL is valid only in the Pulse Width Measurement mode. • A high period measurement is triggered by the rising edge on the reference clock input. • A low period measurement is triggered by the falling edge on the reference clock input. 0b - High period 1b - Low period
8 PULSE_MODE	Pulse Width Measurement Mode Select Selects the measurement mode—either Frequency Measurement mode or Pulse Width Measurement mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	In Frequency Measurement mode, measurement begins at the rising edge of the selected reference clock and continues until a count of 2^{REF_SCALE} reference clock pulses is reached. In Pulse Width Measurement mode, the counter starts incrementing when the selected trigger edge (rising edge for high period measurement or falling edge for low period) occurs, and stops at the next edge (falling edge for a high period measurement or rising edge for a low period measurement). Select high or low period measurement by writing the desired value to PULSE_POL. 0b - Frequency Measurement mode 1b - Pulse Width Measurement mode
7-5 —	Reserved
4-0 REF_SCALE	Reference Clock Scaling Factor Specifies the reference clock scaling factor in Frequency Measurement mode. The reference count cycle is 2^{REF_SCALE}. A higher number provides better accuracy but consumes more processing time. This field is valid only in Frequency Measurement mode.

58.6.1.4 Control Status (CTRLSTAT)

Offset

Register	Offset
CTRLSTAT	4h

Function

Contains the current CTRL_W register configuration fields.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MEASURE...	CONTINU...	0			RESULT_...	GT_MAX_...	LT_MIN_...	0							
W						W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RESULT_...	GT_MAX_...	LT_MIN_...	0		PULSE_P...	PULSE_M...	0			REF_SCALE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 MEASURE_IN_PROGRESS	Measurement in Progress Status Indicates the current CTRL_W[MEASURE_IN_PROGRESS] value. 0b - Not in progress 1b - In progress
30 CONTINUOUS_MODE_EN	Continuous Mode Enable Status Indicates the current CTRL_W[CONTINUOUS_MODE_EN] value. 0b - Disabled 1b - Enabled
29-27 —	Reserved
26 RESULT_READY_STAT	Result Ready Status Indicates that a measurement is complete and CTRL_R[RESULT] is ready to read. Write 1 to this field to clear. 0b - Not complete 1b - Complete
25 GT_MAX_STAT	Greater Than Maximum Result Status Indicates that a measurement is complete and the result is greater than the maximum expected value (CTRL_R[RESULT] > MAX[MAX_VALUE]). Write 1 to this field to clear. 0b - Less than MAX[MAX_VALUE] 1b - Greater than MAX[MAX_VALUE]
24 LT_MIN_STAT	Less Than Minimum Results Status Indicates that a measurement is complete and the result is less than the minimum expected value (CTRL_R[RESULT] < MIN[MIN_VALUE]). Write 1 to this field to clear. 0b - Greater than MIN[MIN_VALUE] 1b - Less than MIN[MIN_VALUE]
23-15 —	Reserved
14 RESULT_READY_INT_EN	Result Ready Interrupt Enable Indicates the current CTRL_W[RESULT_READY_INT_EN] value. 0b - Disabled 1b - Enabled
13	Greater Than Maximum Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
GT_MAX_INT_EN	Indicates the current CTRL_W[GT_MAX_INT_EN] value. 0b - Disabled 1b - Enabled
12 LT_MIN_INT_EN	Less Than Minimum Interrupt Enable Indicates the current CTRL_W[LT_MIN_INT_EN] value. 0b - Disabled 1b - Enabled
11-10 —	Reserved
9 PULSE_POL	Pulse Polarity Indicates the current CTRL_W[PULSE_POL] value. 0b - High period 1b - Low period
8 PULSE_MODE	Pulse Mode Indicates the current CTRL_W[PULSE_MODE] value. 0b - Frequency Measurement mode 1b - Pulse Width Measurement mode
7-5 —	Reserved
4-0 REF_SCALE	Reference Scale Indicates the current CTRL_W[REF_SCALE] value.

58.6.1.5 Minimum (MIN)

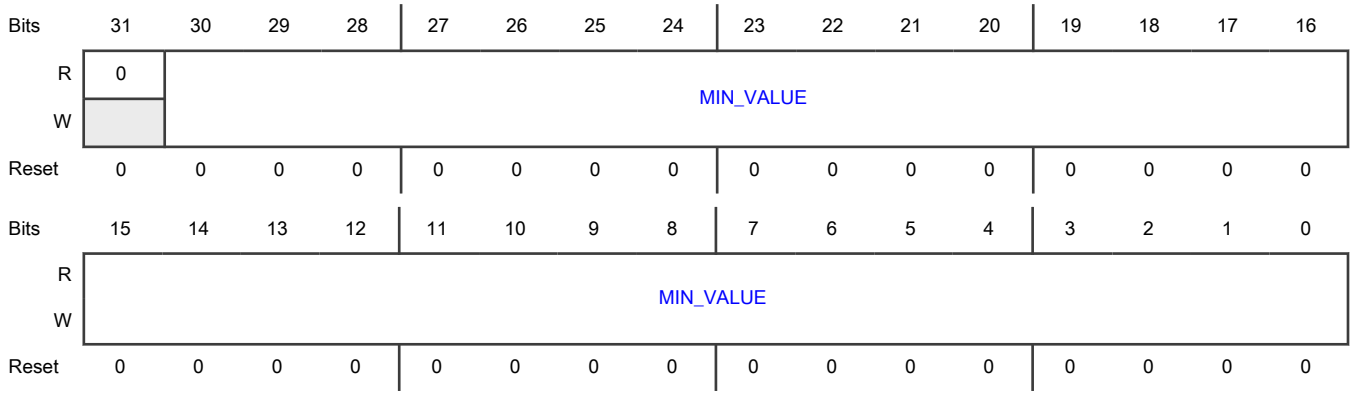
Offset

Register	Offset
MIN	8h

Function

Specifies the minimum expected value for the measurement result.

Diagram



Fields

Field	Function
31 —	Reserved
30-0 MIN_VALUE	Minimum Value Minimum expected value for the measurement result.

58.6.1.6 Maximum (MAX)

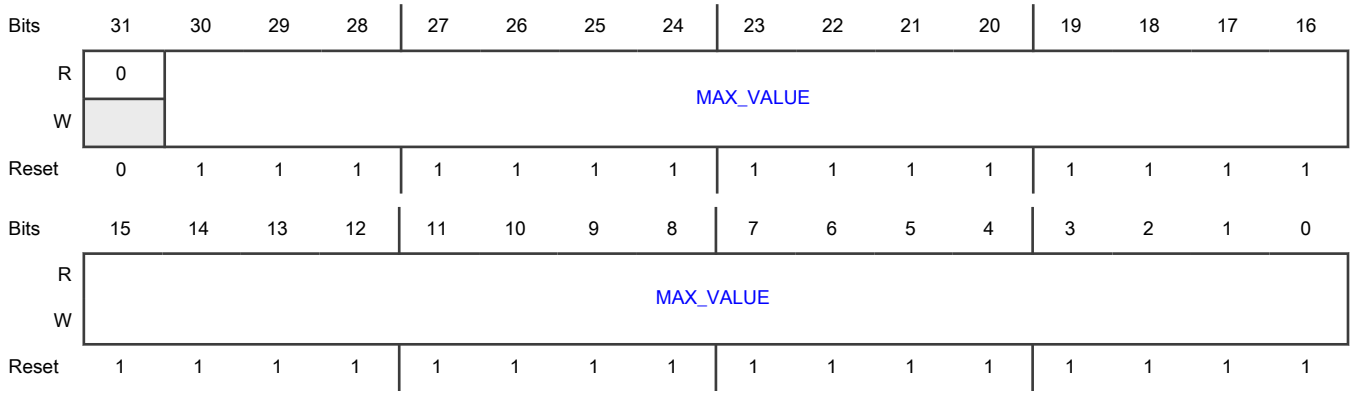
Offset

Register	Offset
MAX	Ch

Function

Specifies the maximum expected value for the measurement result.

Diagram



Fields

Field	Function
31 —	Reserved
30-0 MAX_VALUE	Maximum Value Maximum expected value for the measurement result.

Chapter 59
Low-Power Timer (LPTMR)

59.1 Chip-specific LPTMR information

Table 462. Reference links to related information

Topic	Related module	Reference
Full description	LPTMR	LPTMR
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

59.1.1 Module instances

This device has two instances of the LPTMR module, LPTMR0 and LPTMR1.

59.1.2 LPTMR clocks

The prescaler and glitch filter of the LPTMR module (the module functional clock) can be clocked from one of four sources determined by LPTMRx_PSR[PCS]. The following table shows the clock assignments for this field.

NOTE

The chosen clock must remain enabled if the LPTMR is to continue operating in all required low-power modes. Refer Module operation in low-power modes to see low power modules of LPTMR.

The LPTMR allows the maximum clock frequency of 25 MHz.

Table 463. LPTMRn prescaler/glitch filter clocking options

LPTMRn_PSR[PCS]	Prescaler/glitch filter clock number	Chip clock
00	0	FRO_12M - Free Running Oscillator - 12 MHz clock source.
01	1	FRO_16K - Free Running Oscillator - 16 K clock source.
10	2	32K_CLK - Selectable clock output from the Battery Backed domain produced by either the OSC_32K or the FRO_32K.
11	3	OSC_SYS - Crystal Oscillator - System - Clock frequency must be less than 25 MHz to be used as a clock for LPTMR.

59.1.3 LPTMR pulse counter input options

LPTMRn_CSR[TPS] configures the input source used in pulse counter mode. The following table shows the input assignments for this field.

Table 464. LPTMRn (n=0,1) pulse counter input options

LPTMRn_CSR[TPS]	Pulse counter input number	Chip input
00	0	CMP0_OUT
01	1	CMP1_OUT
10	2	LPTMRn_Alt2
11	3	LPTMRn_Alt3

59.2 Overview

You can configure LPTMR to operate as a time counter with an optional prescaler, or as a pulse counter with an optional glitch filter, across all power modes, including low-power modes. It is reset only on VSYS warm reset, allowing it to be used as a time-of-day counter.

59.2.1 Block diagram

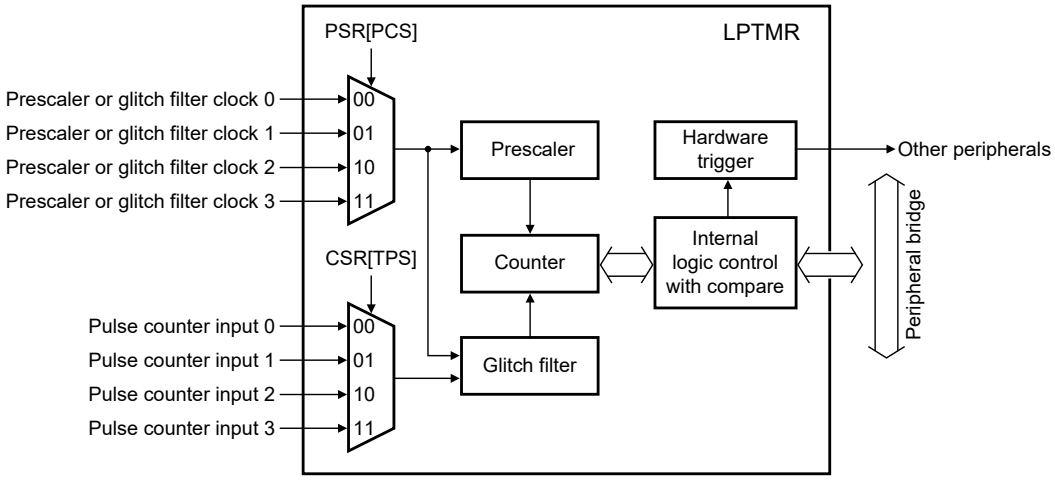


Figure 326. Block diagram

59.2.2 Features

- 32-bit time counter or pulse counter with compare:
  - Optional interrupt that can generate an asynchronous wake-up from any low-power mode
  - Hardware trigger output
  - Counter that supports a free-running mode or reset on compare
- Configurable clock source for prescaler and glitch filter
- Configurable input source for pulse counter (rising-edge or falling-edge)

59.3 Functional description

59.3.1 Low-power modes

In low-power modes, LPTMR continues to operate normally. You can configure LPTMR to exit a low-power mode by generating either an interrupt or a DMA request.

59.3.2 Clocks

The LPTMR prescaler and glitch filter can be clocked by one of the clocks that you configure by using PSR[PCS]. You must enable the clock source before you enable LPTMR.

In Pulse Counter mode, with the glitch filter bypassed, the selected input source directly clocks Counter (CNR), and no other clock source is required. To minimize power in this case, configure the glitch filter clock source for a clock that is disabled.

NOTE

- You may need to configure the clock source that you select in PSR[PCS] for it to remain enabled in low-power modes. Otherwise, LPTMR does not operate in low-power modes.
- The clock source or pulse input source selected for LPTMR must not exceed the maximum frequency of f_{LPTMR} defined in the chip data sheet.

59.3.3 Reset

LPTMR is reset only on VSYS warm reset. When configuring LPTMR registers, you must initially write to Control Status (CSR) with LPTMR disabled, before configuring Prescaler and Glitch Filter (PSR) and Compare (CMR). Then, you must write 1 to CSR[TEN] as the last step in the initialization. Doing so ensures that LPTMR is configured correctly and the LPTMR counter is reset to zero following a VSYS warm reset.

59.3.4 Prescaler and glitch filter

The LPTMR prescaler and glitch filter share the same logic, which operates as a prescaler in Time Counter mode and as a glitch filter in Pulse Counter mode.

NOTE

You must not alter the prescaler and glitch filter configuration when LPTMR is enabled.

59.3.4.1 Prescaler enabled

In Time Counter mode, when the prescaler is enabled, the output of the prescaler directly clocks Counter (CNR). When LPTMR is enabled, CNR increments every 2^1 to 2^{16} prescaler clock cycles. After LPTMR is enabled, the first increment of CNR takes an additional one or two prescaler clock cycles because of the synchronization logic.

59.3.4.2 Prescaler bypassed

In Time Counter mode, when the prescaler is bypassed, the selected prescaler clock increments Counter (CNR) on every clock cycle. When LPTMR is enabled, the first increment takes an additional one or two prescaler clock cycles because of the synchronization logic.

59.3.4.3 Glitch filter enabled

In Pulse Counter mode, when the glitch filter is enabled, the output of the glitch filter directly clocks Counter (CNR). When LPTMR is first enabled, the output of the glitch filter is asserted, that is, logic 1 for active-high and logic 0 for active-low. The following table shows the change in glitch filter output with the selected input source.

Table 465. Glitch filter output with the selected input source

If	Then
The selected input source remains deasserted for at least 2^1 to 2^{15} consecutive prescaler clock rising edges	The glitch filter output also deasserts.
The selected input source remains asserted for at least 2^1 to 2^{15} consecutive prescaler clock rising edges	The glitch filter output also asserts.

NOTE

The input is sampled only on the rising clock edge.

The value of CNR increments each time the glitch filter output asserts. In Pulse Counter mode, the maximum rate at which CNR can increment is once every 2^2 to 2^{16} glitch filter clock edges. When first enabled, the glitch filter waits for an additional one or two glitch filter clock edges because of the synchronization logic.

59.3.4.4 Glitch filter bypassed

In Pulse Counter mode, when the glitch filter is bypassed, the selected input source increments the value of Counter (CNR) every time it asserts. Before LPTMR is first enabled, the selected input source is forced to be asserted. This prevents CNR from incrementing if the selected input source is already asserted when LPTMR is first enabled.

59.3.5 Counter

The value of Counter (CNR) increments by 1 on every:

- Prescaler clock in Time Counter mode, with prescaler bypassed.
- Prescaler output in Time Counter mode, with prescaler enabled.
- Input source assertion in Pulse Counter mode, with glitch filter bypassed.
- Glitch filter output in Pulse Counter mode, with glitch filter enabled.

CNR is reset when LPTMR is disabled or if the counter register overflows. If CSR[TFC] = 0, then CNR is also reset whenever CSR[TCF] = 1.

When the core is halted in Debug mode:

- CNR continues incrementing if configured for Pulse Counter mode.
- CNR stops incrementing if configured for Time Counter mode.

You cannot initialize CNR but can read it at any time. On each read of CNR, you must first write a value to it. This synchronizes and registers the current value of CNR into a temporary register. The contents of the temporary register are returned on each read of CNR.

When reading CNR, the bus clock must be at least two times faster than the rate at which the LPTMR counter is incrementing; otherwise, incorrect data may be returned.

59.3.6 Compare

After the next Counter (CNR) increment (when its value is equal to that of Compare (CMR)), the following events occur:

- CSR[TCF] is read as 1b.
- LPTMR generates an interrupt if CSR[TIE] is 1 as well.
- LPTMR generates a hardware trigger.
- LPTMR writes 0 to CNR if CSR[TFC] is 0.

When LPTMR is enabled, you can modify the value of CMR only when CSR[TCF] is 1. When updating CMR, you must write to it and clear CSR[TCF] before the LPTMR counter increments past the new LPTMR compare value.

NOTE

When LPTMR is enabled in Time Counter mode, the first increment takes an additional one or two clock cycles because of the synchronization logic. This results in the first compare (and therefore interrupt and hardware trigger) occurring slightly later. A faster prescaler clock or larger prescaler value minimizes this impact.

59.3.7 Interrupt

LPTMR generates an interrupt whenever CSR[TIE] and CSR[TCF] are 1. CSR[TCF] is cleared by disabling LPTMR or writing a logic 1 to it.

You can modify the value of CSR[TIE] and write 1 to CSR[TCF] when LPTMR is enabled.

LPTMR generates an interrupt asynchronously to the system clock. The interrupt can be used to generate a wake-up from any low-power mode, provided LPTMR is enabled as a wake-up source.

59.3.8 Hardware trigger

The LPTMR hardware trigger asserts at the same time CSR[TCF] is set and can be used to trigger hardware events in other peripherals without your intervention. The hardware trigger is always enabled.

Table 466. Hardware trigger

When	Then
CMR[COMPARE] and CSR[TFC] are 0	The LPTMR hardware trigger asserts on the first compare and does not deassert.
CMR[COMPARE] is set to a nonzero value, or if CSR[TFC] = 1	The LPTMR hardware trigger asserts on each compare and deasserts on the following increment of Counter (CNR).

59.4 External signals

Table 467. External signals

Signal	Description		Direction
LPTMR_ALT*n*	Pulse Counter Input LPTMR can select one of the input pins to be used in Pulse Counter mode.		Input
	State meaning	Assertion—If configured for Pulse Counter mode with an active-high input, assertion causes Counter (CNR) to increment. Deassertion—If configured for Pulse Counter mode with an active-low input, deassertion causes CNR to increment.	
	Timing	Assertion or deassertion may occur at any time; input may assert asynchronously to the bus clock.	

59.5 Initialization

Perform the following procedure to initialize LPTMR:

1. Configure Control Status (CSR) for the selected mode and pin configuration, when CSR[TEN] is 0. This resets the counter and clears the flag.
2. Configure Prescaler and Glitch Filter (PSR) with the selected clock source and prescaler or glitch filter configuration.
3. Configure Compare (CMR) with the selected compare point.
4. Write 1 to CSR[TEN] to enable LPTMR.

59.6 Application information

59.6.1 Application 1: Generate an interrupt every 100 ms using 32.768 kHz clock source

1. Disable LPTMR by writing 0 to CSR[TEN].
2. Select a 32.768 kHz clock source by configuring PSR[PCS].
3. Bypass the prescaler and glitch filter by writing 1 to PSR[PBYP].
4. Assert an interrupt every 3277 cycles by configuring CMR[COMPARE] = 0CCCh.
5. Enable LPTMR by writing 1 to CSR[TEN].
6. Enable the LPTMR interrupt by writing 1 to CSR[TIE].

NOTE

This is just an example. See the chip-specific LPTMR information for the clocks supported on a given chip.

59.6.2 Application 2: Generate an interrupt once a minute using 32.768 kHz clock source

1. Disable LPTMR by writing 0 to CSR[TEN].
2. Select a 32.768 kHz clock source by configuring PSR[PCS].
3. Select the prescaler to divide the prescaler clock by 32768 to increment the counter once a second by configuring PSR[PRESCALE] = 0Eh.
4. Assert an interrupt every 60 seconds by configuring CMR[COMPARE] = 003Bh.
5. Enable LPTMR by writing 1 to CSR[TEN].
6. Enable the LPTMR interrupt by writing 1 to CSR[TIE].

NOTE

This is just an example. See the chip-specific LPTMR information for the clocks supported on a given chip.

59.7 Memory map and register definition

NOTE

The LPTMR registers are reset only on VSYS warm reset. See Reset for more information.

59.7.1 LPTMR register descriptions

59.7.1.1 LPTMR memory map

LPTMR0 base address: 4004_A000h

LPTMR1 base address: 4004_B000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Status (CSR)	32	RW	0000_0000h
4h	Prescaler and Glitch Filter (PSR)	32	RW	0000_0000h
8h	Compare (CMR)	32	RW	0000_0000h
Ch	Counter (CNR)	32	RW	0000_0000h

59.7.1.2 Control Status (CSR)

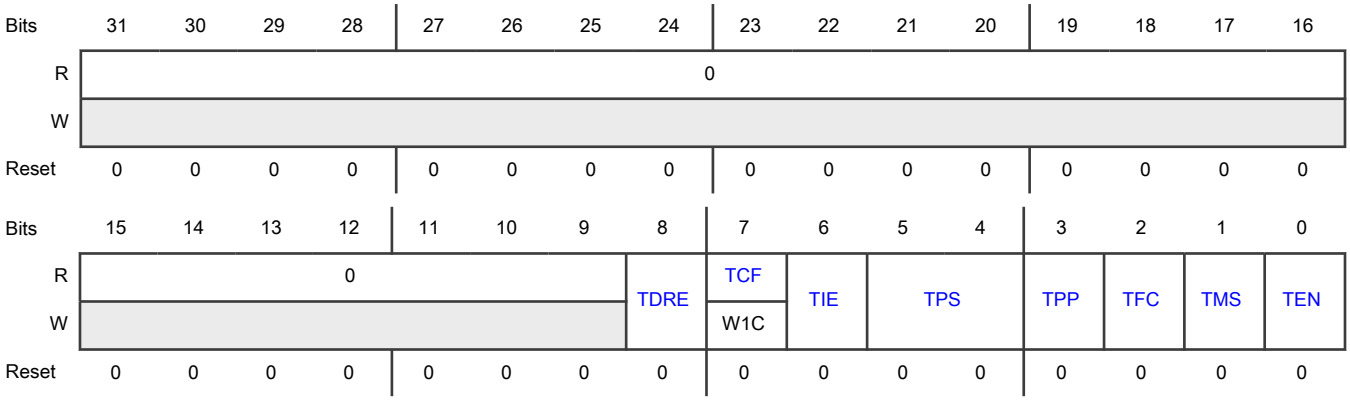
Offset

Register	Offset
CSR	0h

Function

Controls various features of LPTMR.

Diagram



Fields

Field	Function
31-9 —	Reserved
8 TDRE	Timer DMA Request Enable Enables the timer DMA request. When TDRE is 1, the LPTMR DMA request is generated whenever CSR[TCF] is also set. Then, CSR[TCF] is cleared after the DMA controller completes execution. 0b - Disable 1b - Enable
7 TCF	Timer Compare Flag Compares the timer. TCF sets on the next Counter (CNR) increment when LPTMR is enabled and Counter (CNR) equals Compare (CMR). TCF is cleared when LPTMR is disabled or a logic 1 is written to it. **NOTE** You must clear this flag before enabling the timer interrupt or DMA request. **NOTE** This field behaves differently for register reads and writes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When reading 0b - CNR ≠ (CMR + 1) 1b - CNR = (CMR + 1) When writing 0b - No effect 1b - Clear the flag
6 TIE	Timer Interrupt Enable Enables the timer interrupt. If TIE is 1, then LPTMR generates an interrupt if CSR[TCF] is 1 as well. 0b - Disable 1b - Enable
5-4 TPS	Timer Pin Select Configures the input source to be used in Pulse Counter mode. The input connections vary by chip. For details, see the chip configuration information about connections to these inputs. You must modify this field only when LPTMR is disabled. 00b - Input 0 01b - Input 1 10b - Input 2 11b - Input 3
3 TPP	Timer Pin Polarity Configures the polarity of the input source in Pulse Counter mode. If TPP is 0, then the pulse counter input source is active-high, and Counter (CNR) increments on the rising edge. If TPP is 1, then the pulse counter input source is active-low, and CNR increments on the falling edge. You must modify this field only when LPTMR is disabled. 0b - Active-high 1b - Active-low
2 TFC	Timer Free-Running Counter Specifies when the counter resets. If TFC is 0, Counter (CNR) resets on the count cycle following Counter (CNR) becoming equal to Compare (CMR). If TFC is 1, CNR resets on overflow. In both cases, CSR[TCF] sets to 1 on the cycle after CNR and CMR match. You must modify this field only when LPTMR is disabled. 0b - Reset when TCF asserts 1b - Reset on overflow
1	Timer Mode Select Configures the mode of LPTMR.

Table continues on the next page...

Table continued from the previous page...

Field	Function
TMS	You must modify this field only when LPTMR is disabled. 0b - Time Counter 1b - Pulse Counter
0 TEN	Timer Enable Enables the LPTMR timer. If TEN is 0, it resets the LPTMR internal logic, including CNR[COUNTER] and CSR[TCF]. If TEN is 1, LPTMR is enabled. Do not alter CSR[5:1] when writing 1 to this field. 0b - Disable 1b - Enable

59.7.1.3 Prescaler and Glitch Filter (PSR)

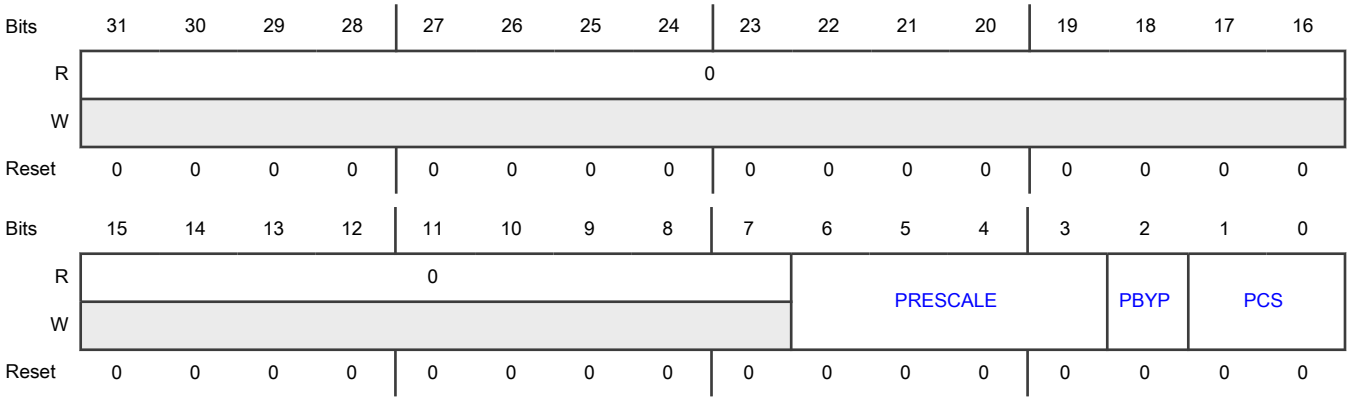
Offset

Register	Offset
PSR	4h

Function

Configures features related to the prescaler and glitch filter.

Diagram



Fields

Field	Function
31-7 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
6-3 PRESCALE	Prescaler and Glitch Filter Value Configures the size of the prescaler in Time Counter mode and the width of the glitch filter in Pulse Counter mode. The width of the glitch filter can vary by one cycle because of the pulse counter input synchronization. You must modify this field only when LPTMR is disabled. 0000b - Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration 0001b - Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after two rising clock edges 0010b - Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after four rising clock edges 0011b - Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after eight rising clock edges 0100b - Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges 0101b - Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges 0110b - Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges 0111b - Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges 1000b - Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges 1001b - Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges 1010b - Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges 1011b - Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges 1100b - Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges 1101b - Prescaler divides the prescaler clock by 16,384; glitch filter recognizes change on input pin after 8192 rising clock edges 1110b - Prescaler divides the prescaler clock by 32,768; glitch filter recognizes change on input pin after 16,384 rising clock edges 1111b - Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges
2 PBYP	Prescaler and Glitch Filter Bypass Controls the clocking of Counter (CNR). If PBYP is 0, the output of the prescaler or glitch filter clocks CNR. If PBYP is 1, the selected prescaler clock in Time Counter mode, or else the selected input source in Pulse Counter mode, directly clocks CNR.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	You must modify this field only when LPTMR is disabled. 0b - Prescaler and glitch filter enable 1b - Prescaler and glitch filter bypass
1-0 PCS	Prescaler and Glitch Filter Clock Select Selects the clock to be used by the LPTMR prescaler and glitch filter. In Time Counter mode, PCS selects the input clock to the prescaler. In Pulse Counter mode, PCS selects the input clock to the glitch filter. See the chip configuration details for information on connections to these inputs. You must modify this field only when LPTMR is disabled. 00b - Clock 0 01b - Clock 1 10b - Clock 2 11b - Clock 3

59.7.1.4 Compare (CMR)

Offset

Register	Offset
CMR	8h

Function

Configures the compare values to Counter (CNR) (see Compare for more information).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COMPARE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMPARE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 COMPARE	Compare Value Configures the compare values to Counter (CNR). On the next CNR increment, if LPTMR is enabled and Counter (CNR) equals Compare (CMR), then: 1. LPTMR writes 1 to CSR[TCF]. 2. The hardware trigger asserts until the next time CNR increments. If CMR = 0, the hardware trigger remains asserted until LPTMR is disabled. If LPTMR is enabled, you must modify the value of CMR only if CSR[TCF] is 1.

59.7.1.5 Counter (CNR)

Offset

Register	Offset
CNR	Ch

Function

Specifies counter values (see Counter for more information).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 COUNTER	Counter Value Contains the current value of the LPTMR counter at the time you last wrote to this register.

Chapter 60
External Watchdog Monitor (EWM)

60.1 Chip-specific EWM information

Table 468. Reference links to related information

Topic	Related module	Reference
Full description	EWM	EWM
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

60.1.1 Module instances

This device has one instance of the EWM module, EWM0.

60.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

60.1.3 CLKPRESCALER register implementation

The CLKPRESCALER register is used for prescaling the clock frequency of low power clock source where:

Prescaled clock frequency = low power clock source frequency / (1 + CLK_DIV).

60.1.4 Low power clock sources

For this device, the EMW's LPO_CLK options are connected as shown in the table below. See EWM clocking for details.

Table 469. LPO_CLK connections

EWM LPO_CLK	SoC clock
LPO_CLK[0]	EWM LPO CLK
LPO_CLK[1]	Not connected/Do not use
LPO_CLK[2]	Not connected/Do not use
LPO_CLK[3]	Not connected/Do not use

60.2 Overview

For safety purposes, a redundant watchdog system, EWM, is designed to monitor external circuits and the MCU software flow. This provides a backup mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

The internal watchdog is used to monitor the flow and execution of the embedded software within the MCU. It consists of a counter that, if allowed to overflow, forces an internal, asynchronous reset to all on-chip peripherals. The counter also optionally asserts the RESET_B pin to reset external devices and circuits. The watchdog counter must not overflow if the software code works well and services the watchdog to restart the actual counter.

The EWM does not reset the MCU's CPU and peripherals, making it different from internal watchdog. The EWM module provides an independent ewm_out_b signal that, when asserted, resets or places an external circuit into a safe mode. The ewm_out_b signal asserts upon EWM counter timeout. An optional external input, ewm_in, allows additional control when asserting the ewm_out_b signal.

60.2.1 Block diagram

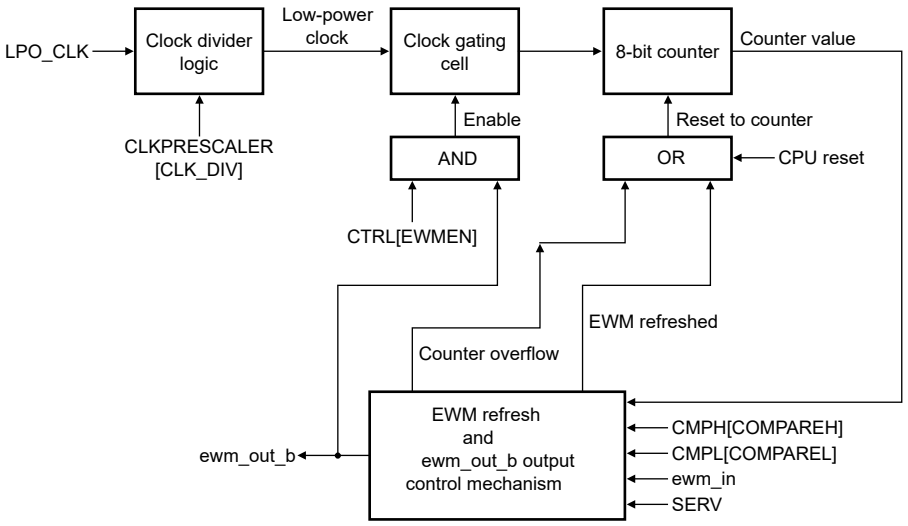


Figure 327. Block diagram

60.2.2 Features

- Independent LPO_CLK source
- Programmable timeout period, specified in terms of the number of EWM LPO_CLK cycles
- Windowed refresh option that provides:
  - A robust check to confirm that the program flow is faster than expected.
  - A programmable window.
  - Refresh outside the window, leading to assertion of the ewm_out_b signal.
- Robust refresh mechanism:
  - Write values of B4h and 2Ch to Service (SERV) within 15 peripheral bus clock cycles.
- One output port, ewm_out_b, which when asserted is used to reset or place the external circuit into Safe mode
- One input port, ewm_in, which allows an external circuit to control the assertion of the ewm_out_b signal

60.3 Functional description

The following sections discuss these aspects of EWM:

- Functional details
- Operating modes

NOTE

If the BUS_CLK is lost, EWM does not generate the ewm_out_b signal and no refresh operation is possible.

60.3.1 Modes of operation

60.3.1.1 Stop mode

When EWM is in Stop mode, the CPU cannot refresh EWM. After entering Stop mode, the EWM counter freezes.

Following are the possible ways to exit Stop mode:

- Through a reset: EWM remains disabled in this case.
- Through an interrupt: EWM is re-enabled and the counter continues to be clocked from the same value as prior to Stop mode entry.

NOTE

Consider the following if EWM enters Stop mode during the CPU refresh mechanism:

- While exiting Stop mode through an interrupt, the refresh mechanism starts from the previous state. That is, if you write the refresh command correctly and EWM enters Stop mode immediately, you must write the next command in *15* peripheral bus clocks after exiting Stop mode.
- You must mask all interrupts before executing the EWM refresh instructions.

60.3.1.2 Debug mode

EWM remains unimpacted when entering Debug mode:

- If EWM is enabled before entering Debug mode, it remains enabled.
- If EWM is disabled before entering Debug mode, it remains disabled.

60.3.2 Using the EWM counter

EWM uses an 8-bit ripple counter that is fed by a clock source independent of the peripheral bus clock source. As the preferred timeout is between 1 ms and 100 ms, the actual clock source must be in the kHz range.

The counter is reset to 0 in these conditions:

- After CPU reset
- After the EWM refresh action completes
- At counter overflow

The CPU cannot access the counter value.

60.3.3 Using compare registers

You can write to Compare Low (CMPL) and Compare High (CMPH) only once after a CPU reset and you cannot modify them until another CPU reset occurs. These registers are used to create a refresh window for the EWM module.

You cannot program Compare Low (CMPL) and Compare High (CMPH) with the same value. In case of any attempt, the ewm_out_b signal asserts as soon as the counter reaches the value of Compare Low (CMPL) + 1.

NOTE

- You must update Compare Low (CMPL) and Compare High (CMPH) before enabling EWM. Therefore, you must provide a reasonable time after POR for the external monitoring circuit to stabilize. You must also ensure that the ewm_in pin is deasserted.
- Service should be requested after 1 clock period of slowest clock frequency while updating Compare Low (CMPL) register.

60.3.4 Using the refresh mechanism

Other than the initial configuration of EWM, the CPU can access EWM only through Service (SERV). The CPU must access this register by correctly writing unique data within the windowed time frame, as determined by Compare Low (CMPL) and Compare High (CMPH) for the correct EWM refresh operation. The following table describes conditions that exist and the refresh mechanisms that apply to those conditions.

Table 470. Refresh mechanisms

Condition	Mechanism
The EWM refresh action completes when the value of Compare Low (CMPL) ≤ the counter value ≤ the value of Compare High (CMPH).	The software behaves as expected and the EWM counter resets to 0. The ewm_out_b output signal remains in Deasserted state if, during the EWM refresh action, the ewm_in input is in Deasserted state.
The EWM refresh action completes when the counter value < the value of Compare Low (CMPL).	The software refreshes EWM before the windowed time frame, the counter resets to 0, and the ewm_out_b output signal asserts no matter what the value of the ewm_in input signal is.
The counter value becomes greater than the value of Compare High (CMPH) prior to completion of the EWM refresh action.	The software does not refresh EWM. The EWM counter resets to 0 and the ewm_out_b output signal asserts no matter what the value of the ewm_in input signal is.

See Service (SERV) for more on the refresh mechanism.

60.3.5 Interrupt

When the ewm_out_b signal asserts, an interrupt request can be generated to indicate the assertion of the EWM reset out signal. The interrupt is enabled when CTRL[INTEN] = 1. Writing 0 to this field clears the interrupt request but does not affect the ewm_out_b signal, which can be deasserted only by forcing a system reset.

60.3.6 Clocking

The following table shows EWM clocks.

Table 471. EWM Clocks

Clock	Description
IPG_CLK	This is the system clock and should be turned on for EWM to be able to work properly. During low power modes in which the core is powered down, this clock is disabled,
IPG_CLK_S	This is the IPS clock and is synchronous with IPG_CLK. It is disabled except during IPS write accesses. it is enabled with EWM's IPS_MODULE_EN
LPO_CLOCK[3:0]	EWM can have 4 different clock sources for running its EWM counter and one of them can be selected by EWM_CLKCTRL [CLK_SEL]. This clock is gated when EWM is disabled or when EWM_out is asserted.

60.3.7 Selecting the EWM counter clock

You can program CLKCTRL[CLKSEL] to select from the available low-power clock sources for the EWM counter.

60.3.8 Using the counter clock prescaler

You can program CLKPRESCALER[CLK_DIV] to divide the EWM counter clock source. This divided clock is used to run the EWM counter.

NOTE

The divided clock used to run the EWM counter must not exceed half the frequency of the bus clock.

60.4 External signals

EWM includes external signals and internal options for the counter clock sources, as shown in the following table.

NOTE

All active-low signals are represented with the suffix "_b" throughout the chapter.

Table 472. Signal descriptions

Signal	Description	I/O
ewm_in	EWM's input for the safety status of external safety circuits. You can program the polarity of ewm_in by using CTRL[ASSIN]. The default polarity is active-low.	I
ewm_out_b	EWM's reset out signal	O
lpo_clk[3:0]	Low-power clock sources for the running counter	I

60.4.1 Using the ewm_out_b signal

The ewm_out_b signal is a digital output signal used to gate an external circuit (application-specific) that controls critical safety functions.

The ewm_out_b signal remains deasserted when the CPU regularly refreshes EWM within the programmable refresh window, indicating that the application code is executing as expected.

The ewm_out_b signal asserts in any of the following conditions:

- An EWM refresh action occurs when the counter value is less than the value of Compare Low (CMPL).
- The EWM counter value becomes greater than the value of Compare High (CMPH) and no EWM refresh occurs.
- The functionality of the ewm_in pin is enabled and the ewm_in pin asserts when refreshing EWM.
- After any reset.

The ewm_out_b signal asserts after any reset by the virtue of the external pulldown mechanism on the ewm_out_b pin. To deassert the ewm_out_b signal, write 1 to CTRL[EWMEN] to enable EWM.

If the ewm_out_b signal shares its pad with a digital I/O pin, this actual pad defers to being an input signal on reset. The ewm_out_b signal controls the pad state only after CTRL[EWMEN] enables EWM.

NOTE

The ewm_out_b pad must be in Pulldown state when the EWM functionality is being used and EWM is under reset.

60.4.2 Using the ewm_out_b pin state in low-power modes

During Wait, Stop, and Power-Down modes, the ewm_out_b pin enters a high-impedance state. You have the option to control the logic state of the pin by using an external pull device or by configuring the internal pull device. When the CPU enters Run mode from Wait or Stop mode recovery, the pin resumes its previous state before entering Wait or Stop mode. When the CPU enters Run mode after exiting Power-Down mode, the pin returns to its reset state.

60.4.3 Using the ewm_in signal

The ewm_in signal is a digital input signal for the safety status of external safety circuits. This signal allows an external circuit to control the assertion of the ewm_out_b signal. For example, in the application, an external circuit monitors a critical safety function, and if there is a fault with the safety function, the external circuit can actively initiate the ewm_out_b signal, which controls the gating circuit.

The ewm_in signal is ignored if EWM is disabled, or if CTRL[INEN] = 0 after any reset.

After you enable EWM (by writing 1 to CTRL[EWMEN]) and the ewm_in functionality (by writing 1 to CTRL[INEN]), the ewm_in signal must be in Deasserted state before the CPU starts refreshing EWM. This ensures that the ewm_out_b signal stays in Deasserted state; otherwise, the ewm_out_b output signal asserts.

60.5 Memory map and register definitions

This section contains the module memory map and registers.

NOTE

EWM supports only 8-bit register accesses; 16-bit and 32-bit accesses are not supported.

60.5.1 EWM register descriptions

60.5.1.1 EWM memory map

EWM0 base address: 400C_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CTRL)	8	RW	00h
1h	Service (SERV)	8	RW	00h
2h	Compare Low (CMPL)	8	RW	00h
3h	Compare High (CMPH)	8	RW	FFh
4h	Clock Control (CLKCTRL)	8	RW	00h
5h	Clock Prescaler (CLKPRESCALER)	8	RW	00h

60.5.1.2 Control (CTRL)

Offset

Register	Offset
CTRL	0h

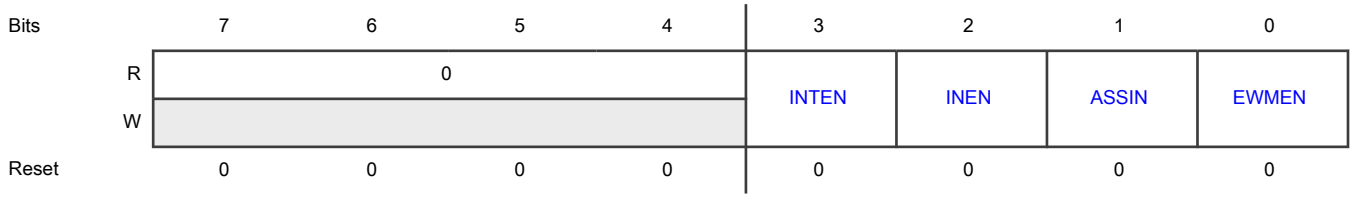
Function

Controls the functionality of EWM.

NOTE

You can write to CTRL[INEN], CTRL[ASSIN], and CTRL[EWMEN] only once after a CPU reset. Modifying these fields more than once generates a bus transfer error.

Diagram



Fields

Field	Function
7-4 —	Reserved
3 INTEN	Interrupt Enable Enables interrupt request generation. If this field = 1 and the ewm_out_b signal is asserted, an interrupt request is generated. To deassert interrupt requests, write 0 to this field. 0b - Deasserts interrupt requests 1b - Generates interrupt requests
2 INEN	Input Enable Enables the ewm_in port. When this field = 1, it enables the ewm_in port. 0b - Disables 1b - Enables
1 ASSIN	Assertion State Select Specifies the asserted state of the ewm_in signal. By default, the asserted state of the ewm_in signal is logic 0 (active-low), which is when this field = 0. When this field = 1, the ewm_in asserted state is logic 1 (active-high). You can use this field to change the expected polarity of the ewm_in signal. 0b - Logic 0 1b - Logic 1
0 EWMEN	EWM Enable Enables the EWM module. If this field = 1, it enables the EWM module, and if the field = 0, it disables the EWM module. You cannot re-enable this field until the next reset because of its write-once nature. 0b - Disables 1b - Enables

60.5.1.3 Service (SERV)

Offset

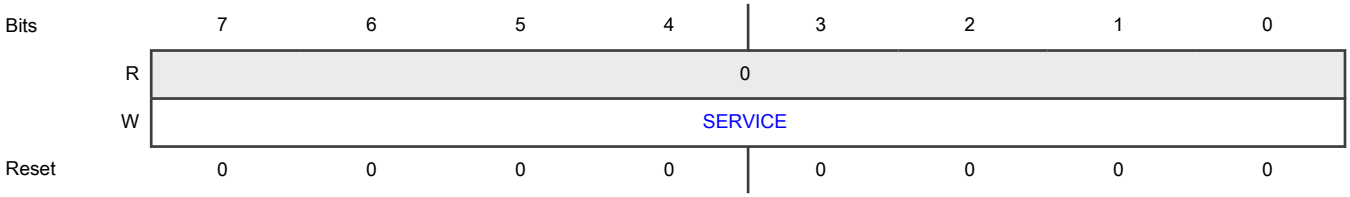
Register	Offset
SERV	1h

Function

Provides an interface from the CPU to the EWM module.

Attempted reads of this register return 0.

Diagram



Fields

Field	Function
7-0 SERVICE	Service Provides an interface from the CPU to the EWM module. The EWM refresh mechanism requires the CPU to write these values to this field: a first data byte of B4h, followed by a second data byte of 2Ch. The EWM refresh action is invalid if either of the following conditions is true: • The first or second data byte is not written correctly. • The second data byte is not written within a fixed number of peripheral bus cycles of the first data byte, known as EWM_refresh_time. The number of peripheral bus clock cycles required for EWM_refresh_time is 15.

60.5.1.4 Compare Low (CMPL)

Offset

Register	Offset
CMPL	2h

Function

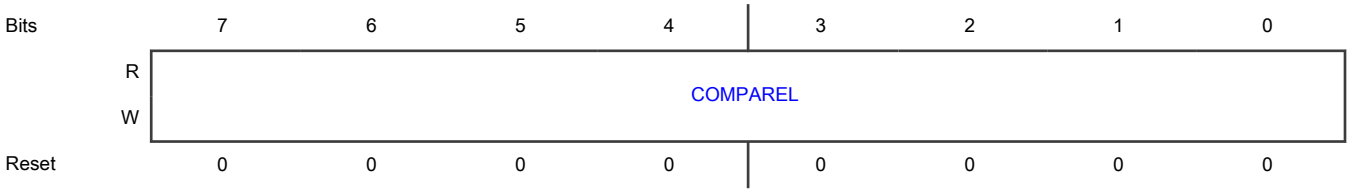
Determines the lower value of the windowed time frame for the correct EWM refresh operation.

This register is reset to 0 after a CPU reset. This provides no minimum time for the CPU to refresh the EWM counter.

NOTE

You can write to this register only once after a CPU reset. Writing to the register more than once generates a bus transfer error.

Diagram



Fields

Field	Function
7-0 COMPAREL	Compare Low Configures the minimum counter value when refreshes are allowed. If a refresh is attempted while the counter value is lower than COMPAREL, then the ewm_out_b signal is asserted. To prevent runaway code from changing the value of this field, you must write to this field after a CPU reset even if the (default) minimum refresh time is required.

60.5.1.5 Compare High (CMPH)

Offset

Register	Offset
CMPH	3h

Function

Determines the higher value of the windowed time frame for the correct EWM refresh operation.

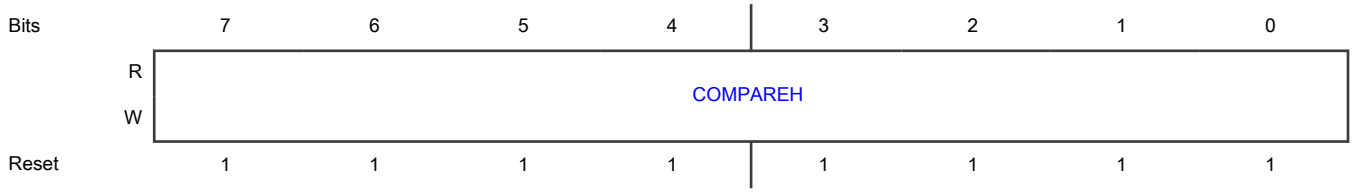
This register is reset to FFh after a CPU reset. This provides a maximum time of up to 256 clocks for the CPU to refresh the EWM counter.

NOTE

You can write to this register only once after a CPU reset. Writing to the register more than once generates a bus transfer error.

The valid values for this register are up to FEh because the EWM counter never expires when the value of COMPAREH = FFh. The expiration happens only if the EWM counter is greater than the value of COMPAREH.

Diagram



Fields

Field	Function
7-0 COMPAREH	Compare High Configures the maximum counter value till when refreshes are allowed. If a refresh is not attempted while the counter value is greater than COMPAREH, then the ewm_out_b signal is asserted. To prevent runaway code from changing the value of this field, you must write to this field after a CPU reset.

60.5.1.6 Clock Control (CLKCTRL)

Offset

Register	Offset
CLKCTRL	4h

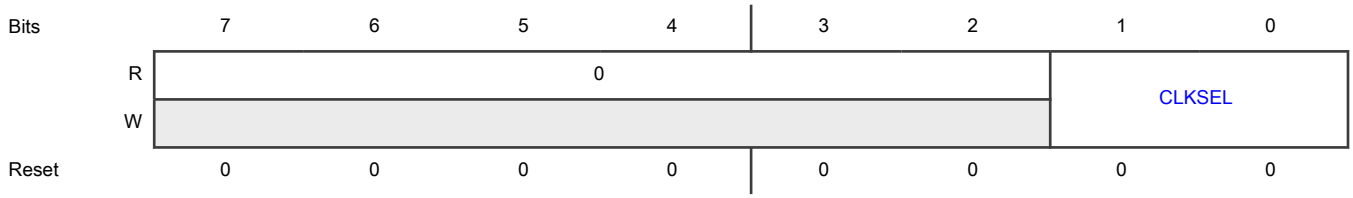
Function

Provides a selection mechanism for low-power clock sources to run the EWM counter.

NOTE

You can write to this register only once after a CPU reset. Writing to the register more than once generates a bus transfer error. You must select the required low-power clock before enabling EWM.

Diagram



Fields

Field	Function
7-2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 CLKSEL	Clock Select Provides a selection mechanism for low-power clock sources to run the EWM counter. EWM has the following available low-power clock sources for running the EWM counter. Write an appropriate value to this field to select one of the clock sources. • 00 - lpo_clk[0] • 01 - lpo_clk[1] • 10 - lpo_clk[2] • 11 - lpo_clk[3] See the chip-specific information for low power clock sources used in your device.

60.5.1.7 Clock Prescaler (CLKPRESCALER)

Offset

Register	Offset
CLKPRESCALER	5h

Function

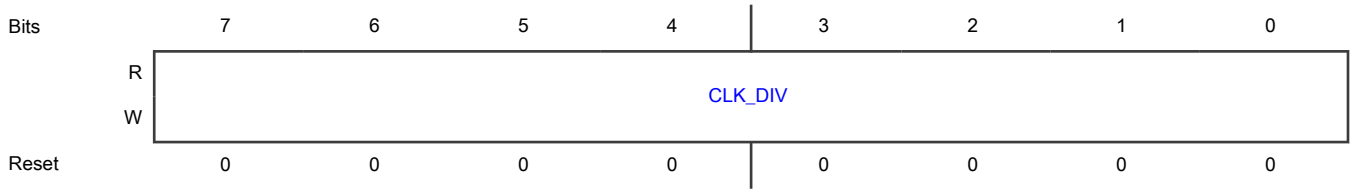
Prescales the EWM counter clock source by a clock divider.

This register is reset to 00h after a CPU reset.

NOTE

You can write to this register only once after a CPU reset. Writing to the register more than once generates a bus transfer error. You must write the required prescaler value before enabling EWM.

Diagram



Fields

Field	Function
7-0 CLK_DIV	Clock Divider Prescales the selected low-power clock source for running the EWM counter:

Table continues on the next page...

Field	Function
	Prescaled clock frequency = low-power clock source frequency ÷ (1 + the value of CLK_DIV) See chip-specific information for low-power clock source frequency used in your device.

Chapter 61
USB Full-Speed Host and Device Controller (USBFS)

61.1 Chip-specific USBFS information

Table 473. Reference links to related information

Topic	Related module	Reference
Full description	USBFS	USBFS
Peripheral memory map		PBRG Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See USBFS_TRIG register in INPUTMUX

NOTE

See AHB Clock Control 1 (AHBCLKCTRL1) to enable USB FS clock. Also see Peripheral Reset Control 1 (PRESETCTRL1) for reset control.

61.1.1 Module instances

This device has one instance of the USBFS module, USBFS0.

61.2 Overview

USBFS comprises an On-The-Go (OTG) dual-role subsystem with OTG protocol support. The OTG implementation in this module provides a device solution for a full-speed compliant peripheral and limited host functionality for implementing a USB 2.0 full-speed/low-speed compliant embedded host. The OTG logic implements features, which *On-The-Go and Embedded Host Supplement to the USB 2.0 Specification* requires.

In this subsystem, the USB full-speed controller interfaces to a USB FS/LS transceiver and Memory map and register definitions also includes configuration controls for that transceiver.

In Device mode, the USBFS subsystem working with the SCG module can use the FIRC to generate a 48 MHz USB controller clock tuned using the incoming host USB data for crystal-less operation. Memory map and register definitions also include the registers necessary for configuration controls and status signals of the USB portion of this crystal-less operation.

61.2.1 Block diagram

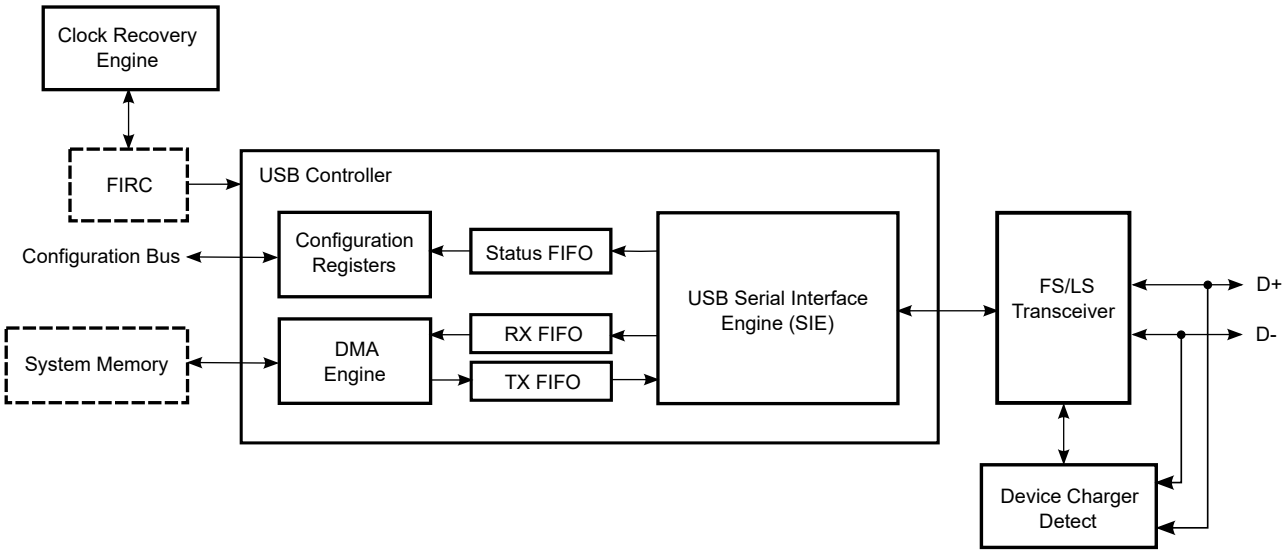


Figure 328. USB FS/LS subsystem overview

61.2.2 Features

- Includes a dual-role USB OTG-capable controller that supports Full-Speed (FS) device or FS/LS embedded host operation. The controller operates with DMA or FIFO data stream interfaces and supports up to 16 bidirectional endpoints. The module complies with the USB 2.0 specification.
- Supports a USB transceiver that includes internal 15 kΩ pulldowns on the D+ and D- lines for Host mode functionality and a 1.5 kΩ pullup on the D+ line for Device mode functionality.
- Supports a USB device charger detection module with USB battery charging 1.2 detection and signaling features (see the "USB DCD" chapter in this Reference Manual for more information).
- Includes FIRC with clock recovery block to eliminate the need for a clock from a 48 MHz crystal. The USB frequency tuning feature is available only for USB device mode operation (see the SCG chapter in this Reference Manual for more information).
- Keep Alive mode allows operation in Sleep mode for certain USB low-traffic situations without needing to restart the USB session after exiting Low-Power mode. In this mode, the USBFS subsystem can respond to IN tokens with NAK replies and initiate wakeup for SETUP or software-assigned tokens.
- A configurable connection allows any one of the LPUART transmit and receive pins to be connected to the Full Speed USB physical layer.
- Supports lane reversal feature to allow switching the roles of package USB0_DM and USB0_DP pins providing additional flexibility for PC board layout.

61.2.3 USB interface general overview

The USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The bus allows peripherals to be attached, configured, used, and detached when the host and other peripherals operate. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol.

USB software provides a uniform system view for all application software, hiding implementation details to make application software more portable. It manages the dynamic attaching and detaching of peripherals.

There is only one host in any USB system. The USB interface to the host computer system is referred to as the host controller.

Any system may have multiple USB devices, such as human interface devices, speakers, printers, etc. USB devices present a standard USB interface regarding comprehension, response, and standard capability.

The host initiates transactions to specific peripherals, whereas the device responds to control transactions. The device sends and receives data to and from the host using a standard USB data format. USB 2.0 full-speed/low-speed peripherals operate at 12 Mbit/s or 1.5 Mbit/s.

See the USB 2.0 specification for more information.

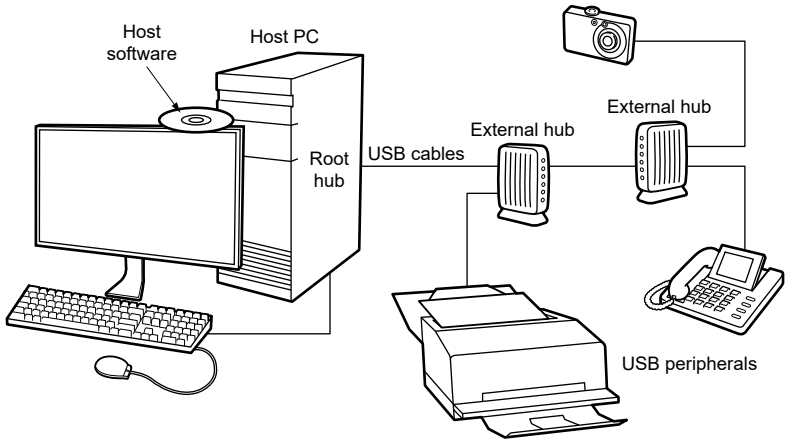


Figure 329. Example USB 2.0 system configuration

61.2.4 USB On-The-Go

USB is a popular standard for connecting peripherals and portable consumer electronic devices such as digital cameras and tablets to host PCs. The OTG Supplement to the USB Specification extends USB to peer-to-peer applications. Using USB OTG technology, consumer electronics, peripherals, and portable devices can connect to exchange data. For example, a digital camera can connect directly to a printer, or a keyboard can connect to a tablet to exchange data.

With the USB OTG product, you can develop a fully USB-compliant peripheral device that can also assume the role of a USB host. The software determines the role of the device based on hardware signals. It then initializes the device in the appropriate mode of operation (host or peripheral) based on how it is connected. After connecting, the devices can negotiate using the OTG protocols to assume the role of host or peripheral based on the task.

See the *On-The-Go and Embedded Host Supplement to the USB 2.0 Specification* for more information.

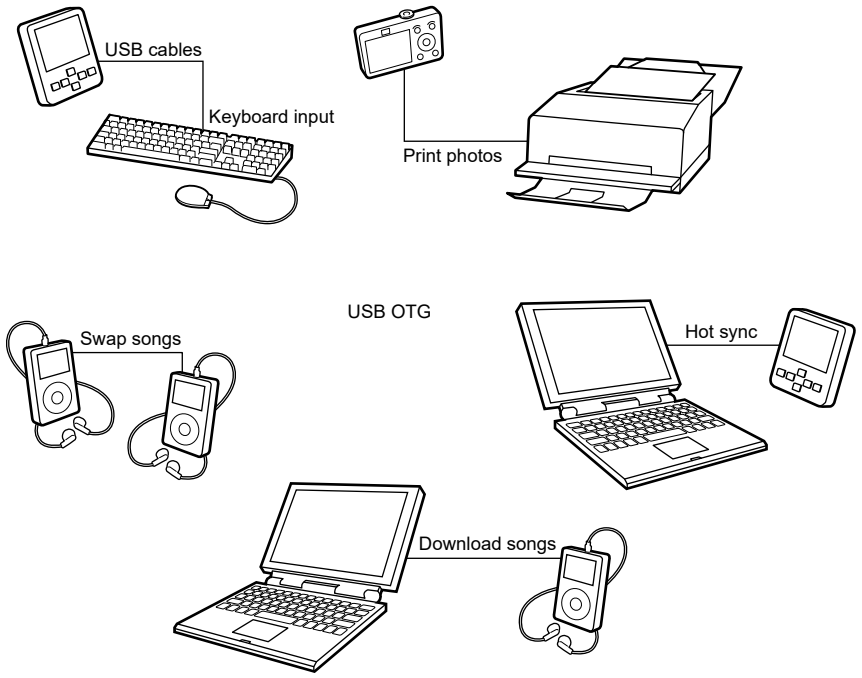


Figure 330. Example USB 2.0 On-The-Go configurations

61.2.5 References

The following publications are referenced in this document. For copies or updates to these specifications, see the USB Implementers Forum, Inc. website at http://www.usb.org.

- *Universal Serial Bus Specification, Revision 2.0*, 2000, with amendments including those listed below
- *Errata for "USB Revision 2.0 April 27, 2000" as of 12/7/2000*
- *Errata for "USB Revision 2.0 April 27, 2000" as of May 28, 2002*
- *Pullup / Pulldown Resistors* (USB Engineering Change Notice)
- *Suspend Current Limit Changes* (USB Engineering Change Notice)
- *Device Capacitance* (USB Engineering Change Notice)
- *USB 2.0 Connect Timing Update* (USB Engineering Change Notice as of April 4, 2013)
- *USB 2.0 VBUS Max Limit* (USB Engineering Change Notice)
- *On-The-Go and Embedded Host Supplement to the USB Revision 2.0 Specification*, Revision 2.0 version 1.1a, July 27, 2012
- *Maximum VBUS Voltage* (USB OTGEH Engineering Change Notice)
- *Universal Serial Bus Micro-USB Cables and Connectors Specification*, Revision 1.01, 2007

61.3 Functional description

61.3.1 On-chip transceiver required external components

USB system operation requires external components to ensure that you have met the driver output impedance, eye diagram, and VBUS cable fault tolerance requirements. DM and DP I/O pads must connect through series resistors (approximately 33 Ω each) to the USB connector on the application printed circuit board (PCB). Additionally, signal quality optimizes when you mount these 33 Ω resistors closer to the processor than the USB board-level connector. The USB transceiver includes the following:

- An internal 1.5 kΩ pullup resistor on the USB_DP line for the full-speed device (CONTROL[DPPULLUPNONOTG] or OTGCTL[DPHIGH] controls).
- Internal 15 kΩ pulldown resistors on the USB_DP and USB_DM signals, primarily intended for Host mode operation, are also useful in Device mode, as explained below.

NOTE

For device operation, you must enable the internal 15 kΩ pulldowns to keep the DP and DM ports in a known quiescent state when the VBUS detection software determines that the USB connection is not activated. It also includes cases when no cable to a host is present or the USB port is not used. USBCTRL[PDE] must control the internal 15 kΩ pulldowns in this case.

For host operation, the internal 15 kΩ pulldowns enable automatically, as the USB 2.0 specification requires when CTL[HOSTMODEEN] is asserted high.

The following diagrams provide overviews of device-only, host-only, and dual-role connections. See *Kinetis Peripheral Module Quick Reference (KQRUG)*, available on www.nxp.com, for more information.

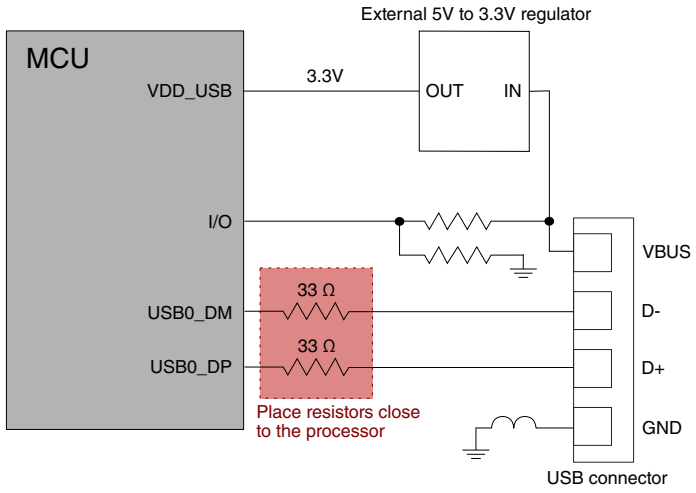


Figure 331. Typical Device-only block diagram (bus-powered with external regulator)

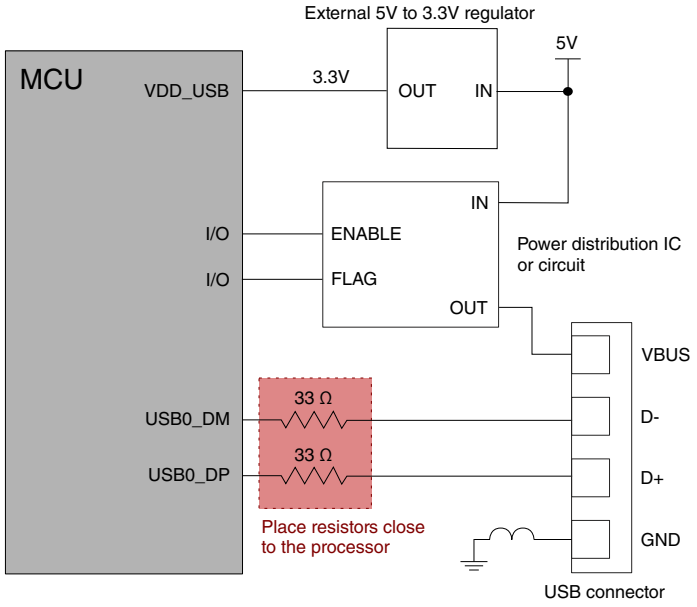


Figure 332. Host-only diagram with external regulator

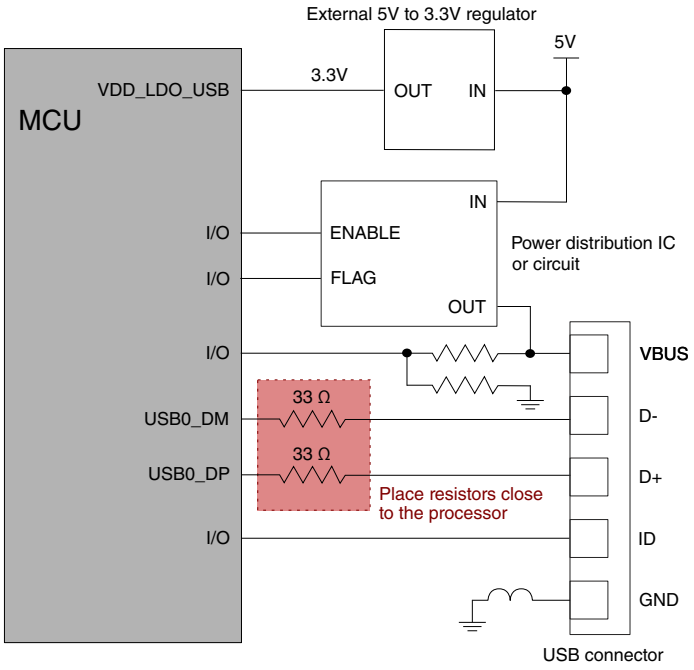


Figure 333. Dual-role diagram with external regulator

61.3.2 Programmer interface

USBFS communicates with the processor core through status registers, control registers, and data structures in memory. This section discusses the major components of the programming model for USBFS.

61.3.2.1 Buffer descriptor table (BDT)

The device operation's function is to transfer a request in the memory image to and from the USB. USBFS implements a BDT to manage USB endpoint communications in system memory efficiently.

- The BDT resides on a 512-byte boundary in system memory and is pointed to by the BDT page registers.
- Every endpoint direction requires two 8-byte buffer descriptor (BD) entries.

Therefore, a system with 16 fully bidirectional endpoints requires 512 bytes of system memory to implement the BDT. The two BD entries allow for an EVEN BD and ODD BD entry for each endpoint direction, which enables the microprocessor to process one BD while USBFS is processing the other BD. Double buffering BDs allow USBFS to transfer data easily at the maximum throughput provided by USB.

You must manage buffers for USBFS by updating the BDT when needed. This allows USBFS to efficiently manage data transmission and reception while the microprocessor performs communication overhead processing and other function-dependent applications.

Because the buffers are shared between the microprocessor and USBFS, a simple semaphore mechanism distinguishes who is allowed to update the BDT and buffers in system memory. A semaphore (the OWN field) becomes 0 when the microprocessor owns the BD entry.

- When OWN = 0, the microprocessor is allowed read and write access to the BD entry and the buffer in system memory.
- When OWN = 1, USBFS owns the BD entry and the buffer in system memory. USBFS now has full read and write access, and the microprocessor must not modify the BD or its corresponding data buffer.

The BD also contains indirect address pointers where the actual buffer resides in system memory. The following diagram shows this indirect address mechanism.

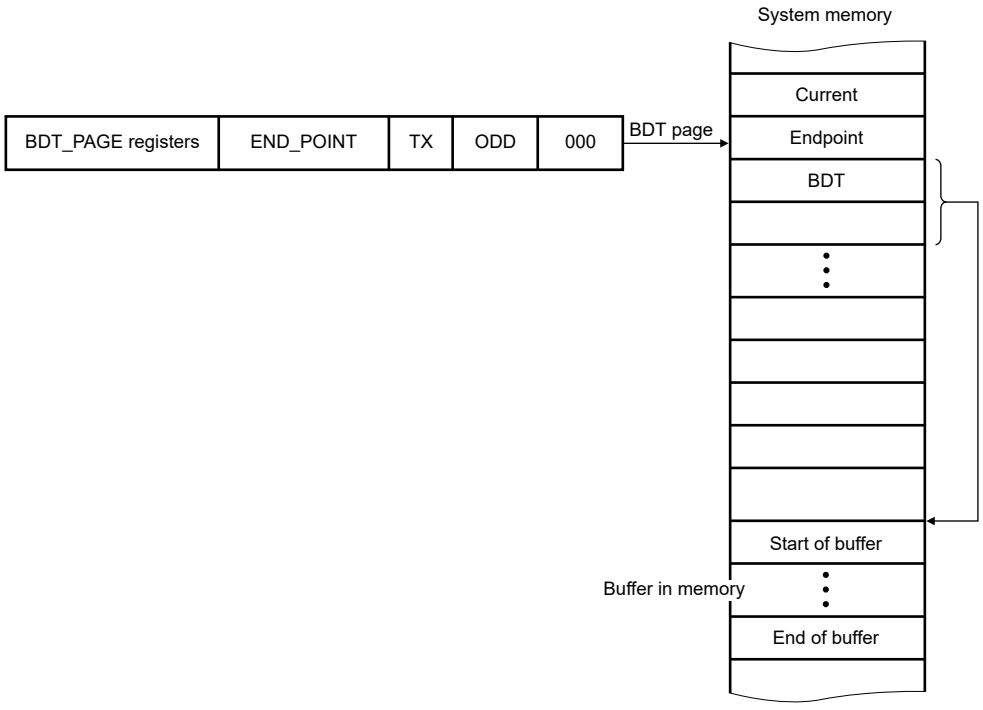


Figure 334. Buffer descriptor table

61.3.2.2 Addressing BDT entries

An understanding of the addressing mechanism of BDT is useful when accessing endpoint data via USBFS or microprocessor:

- The BDT occupies up to 512 bytes of system memory.
- You can support 16 bidirectional endpoints with a full BDT of 512 bytes.
- 16 bytes are needed for each USB endpoint direction.
- Applications with fewer than 16 endpoints require less RAM to implement the BDT.
- The BDT page registers (BDT_PAGE) point to the starting location of the BDT.
- You must locate the BDT on a 512-byte boundary in system memory.
- All enabled transit and receive endpoint BD entries are indexed into the BDT to allow easy access via USBFS or MCU core.

When you receive a USB token on an enabled endpoint, USBFS uses its integrated DMA controller to interrogate the BDT. USBFS reads the corresponding endpoint BD entry to determine whether it (USBFS) owns the BD and corresponding buffer in system memory.

To compute the entry point into the BDT, the BDT_PAGE registers are concatenated with the current endpoint and the TX and ODD fields to form a 32-bit address. This following tables show this address mechanism:

Table 474. BDT address calculation

31:24	23:16	15:9	8:5	4	3	2:0
BDT_PAGE_03	BDT_PAGE_02	BDT_PAGE_01[7:1]	Endpoint	TX	ODD	000

Table 475. BDT address calculation fields

Field	Description
BDT_PAGE	BDT_PAGE registers in the control register block
ENDPOINT	ENDPOINT field from the USB TOKEN
TX	1 for transmit transfers; 0 for receive transfers
ODD	It corresponds to the buffer currently in use; the buffers are used in a ping-pong fashion. Maintained within the USBFS SIE.

61.3.2.3 Buffer descriptors (BDs)

A buffer descriptor provides endpoint buffer control information for USBFS and the processor. The BDs have different meanings, depending on whether USBFS or the processor reads the BD in memory.

The USBFS controller uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- Whether to release ownership upon packet completion
- No address increment (FIFO mode)
- Whether data toggle synchronization is enabled
- How much data to transmit or receive
- Where the buffer resides in system memory

The processor uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- The received TOKEN PID
- How much data was transmitted or received
- Where the buffer resides in system memory

The format for the BD is shown in the following table.

Table 476. Buffer descriptor format

31:26	25:16	15:8	7	6	5	4	3	2	1	0
RSVD	BC (10 bits)	RSVD	OWN	DATA0/1	KEEP/ TOK_PID[3]	NINC/ TOK_PID[2]	DTS/ TOK_PID[1]	BDT_STALL/ TOK_PID[0]	0	0
Buffer Address (32 bits)										

Table 477. Buffer descriptor fields

Field	Description
31–26 RSVD	Reserved
25–16 BC	Byte Count Represents the 10-bit byte count. USBFS SIE changes this field upon completing a receive transfer with the byte count of the data received.
15–8 RSVD	Reserved
7 OWN	Determines whether the processor or USBFS currently owns the buffer. Except when KEEP = 1, the SIE hands ownership back to the processor after completing the token by writing 0 to this field. This must always be the last byte of the BD that the processor updates when it initializes a BD. 0 The processor has access to the BD. USBFS ignores all other fields in the BD. 1 USBFS has access to the BD. When USBFS owns the BD, the processor must not modify any other fields in the BD.
6 DATA0/1	Defines whether you transmit or receive a DATA0 field (DATA0/1=0) or a DATA1 (DATA0/1=1) field. It is unchanged by USBFS.
5 KEEP/ TOK_PID[3]	Has two functions (KEEP or TOK_PID[3]): • KEEP—When written by the processor, it serves as the KEEP bit. Typically, this bit is 1 with ISO endpoints feeding a FIFO. The microprocessor is not informed that a token has been processed. Transfer the data to or from the FIFO. When KEEP becomes 1, the NINC field usually becomes 1 to prevent address increment.

Table continues on the next page...

Table 477. Buffer descriptor fields (continued)

Field	Description
	0 Allows USBFS to release the BD when a token has been processed. 1 This field is unchanged by USBFS. Bit 3 of the current token PID is written back to the BD by USBFS. • TOK_PID[3]—If the OWN field also becomes 1, then the BD remains owned by USBFS indefinitely; when written by USB, it serves as TOK_PID[3]. Typically, this field is 1 with ISO endpoints feeding a FIFO. The microprocessor is not informed that a token has been processed. Transfer the data to or from the FIFO. When KEEP becomes 1, the NINC field usually becomes 1 to prevent address increment. 0 or 1 USBFS writes back bit 3 of the current token PID to the BD.
4 NINC/ TOK_PID[2]	No Increment (NINC) Disables the DMA engine address increment. This forces the DMA engine to read or write from the same address. This is useful for endpoints when data is *read from* or *written to* a single location (such as a FIFO). Typically this field becomes 1 with the KEEP field for ISO endpoints that are interfacing to a FIFO. 0 USBFS writes bit 2 of the current token PID to the BD. 1 This bit is unchanged by USBFS.
3 DTS/ TOK_PID[1]	Enables USBFS to perform data toggle synchronization when you write 1 to this field. • If KEEP = 0, then bit 1 of the current token PID is written back to the BD. • If KEEP = 1, then this field is unchanged by USBFS. 0 Data toggle synchronization is disabled. 1 Enables USBFS to perform data toggle synchronization.
2 BDT_STALL TOK_PID[0]	Causes USBFS to issue a STALL handshake if the SIE receives a token that uses the BDT in this location when you write 1 to this field. The SIE does not consume the BDT (the OWN field remains 1 and the rest of the BDT is unchanged) when a BDT_STALL field becomes 1. • If KEEP = 0, write back bit 0 of the current token PID to the BD. • If KEEP = 1, then this field is unchanged by USBFS. 0 No stall issued. 1 The SIE does not consume the BDT (the OWN field remains 1, and the rest of the BDT is unchanged). Writing 1 to BDT_STALL also causes the corresponding USB_ENDPTn[EPSTALL] to become 1. This causes USB OTG to issue a STALL handshake for both directions of the associated endpoint. To clear the stall condition: 1. Write 0 to the associated USB_ENDPTn[EPSTALL]. 2. Write the BDT to clear OWN and BDT_STALL.
TOK_PID[n]	Bits [5:2] can also represent the current token PID. When a transfer completes, USBFS writes back the current token PID into the BD. The values written back are the token PID values from the USB specification:

Table continues on the next page...

Table 477. Buffer descriptor fields (continued)

Field	Description
	• 1h for an OUT token. • 9h for an IN token. • Dh for a SETUP token. In Host mode, this field reports the last returned PID or a transfer status indication. The possible values returned are: • 3h: DATA0 • Bh: DATA1 • 2h: ACK • Eh: STALL • Ah: NAK • 0h: Bus timeout • Fh: Data error
1–0 Reserved	Reserved
ADDR[31:0]	Address Represents the 32-bit buffer address in system memory; this address must also be 32-bit aligned. USBFS does not change this field.

61.3.2.4 USB transaction

When USBFS transmits or receives data, it computes the BDT address using the address generation shown in Addressing BDT entries.

If OWN = 1, then the following process occurs:

1. USBFS reads the BDT.
2. The SIE transfers the data via the DMA to or from the buffer pointed to by Address (ADDR) of the BD.
3. When the TOKEN completes, USBFS updates the BDT, and if KEEP = 0, then USBFS changes the OWN field to 0.
4. Update the STAT register, and the TOK_DNE interrupt occurs.
5. When the processor processes the TOK_DNE interrupt, it reads from the status register all the information required to process the endpoint.
6. At this point, the processor allocates a new BD so that you can transmit or receive the additional USB data for that endpoint and then processes the last BD.

The following figure shows how a typical USB token is processed after the BDT is read and OWN = 1.

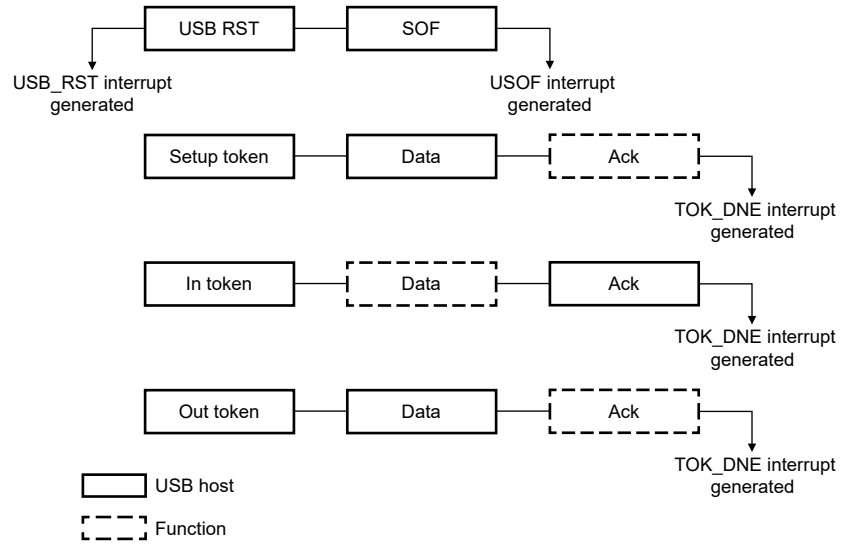


Figure 335. USB token transaction

USB has two sources for the DMA overrun error:

- **Memory latency:** The memory latency may be too high and cause the receive FIFO to overflow. This is predominantly a hardware performance issue, usually caused by transient memory access issues.
- **Oversized packets:** The packet received may be larger than the negotiated MaxPacket size. Typically, a software bug causes this. The USB specification is ambiguous for DMA overrun errors due to oversized data packets because it assumes correct software drivers on both sides. NAKing the packet can result in the retransmission of the already oversized packet data. Therefore, in response to oversized packets, the USB core continues ACKing the packet for non-isochronous transfers.

Table 478. USB responses to DMA overrun errors

Errors due to memory latency	Errors due to oversized packets
Non-acknowledgment (NAK) or bus timeout (BTO) — See bit 4 in Error Interrupt Status (ERRSTAT) as appropriate for the class of transaction.	Continues acknowledging (ACKing) the packet for non-isochronous transfers.
—	Clip the data written to memory to the MaxPacket size so as not to corrupt system memory.
ERRSTAT[DMAERR] becomes 1 for Host mode and Device mode of operation. Depending on the INTENB and ERRENB register's values, the core may assert an interrupt to notify the processor of the DMA error.	Asserts ERRSTAT[DMAERR], which can trigger an interrupt and ISTAT[TOKDNE] fires. The TOK_PID field of the BDT is not 1111 because the DMAERR is not due to latency.
• In Host mode: Generate ISTAT[TOKDNE] and the TOK_PID field of the BDT is 1111 to indicate the DMA latency error. Host mode software can decide to retry or move to next scheduled item. • In Device mode: The BDT is not written back nor ISTAT[TOKDNE] triggered because it is assumed that a second attempt is queued and will succeed in the future.	The packet length field written back to the BDT is the MaxPacket value representing the length of the clipped data written to memory.
From here, you can decide an appropriate course of action for future transactions, such as stalling the endpoint, canceling the transfer, disabling the endpoint, and others.	

61.3.3 Operations

61.3.3.1 Device mode FIRC operation

The following are the FIRC initialization code steps for USB crystal-less Device mode operation:

1. Enable the FIRC nominal 48 MHz clock by ensuring that the FIRCCSR[FIRCEN] and FIRCTCFG[TRIMSRC] fields in the SCG registers have the correct values for crystal-less FS USB Device mode operation.
2. Enable the USB clock recovery tuning by writing 1 to CLK_RECOVER_CTRL[CLOCK_RECOVER_EN].
3. Complete the configuration of the USB clock recovery logic by writing 1 to CLK_RECOVER_IRC_EN[IRC_EN].
4. This chip does not have alternate USB input clock sources from clock muxes, so no additional 48 MHz clock path configuration is required.

61.3.3.2 USB and VREGIN pin status detection

This device does not have a dedicated VBUS detect pin. Still, it does have the capability through CONTROL[SESS_VLD] to report the status of VBUS using one of two monitoring methods with different package pins. The choice between the two VBUS monitoring methods is made with CONTROL[VBUS_SOURCE_SEL]. For VBUS detection, use a specified GPIO pin for both bus-powered and self-powered USB cases. If the GPIO pins on this device do not directly support a 5 V input, use an external resistive voltage divider to keep the input voltage within the valid range if you use the GPIO pin method for VBUS detection. The voltage status of the VDD_LDO_USB pin is passed internally from the USB VREG to the USB FS subsystem, so no additional board connections are required if you choose the VBUS detection method.

For this chip, the specified GPIO package pin for VBUS monitoring is PTB6 configured in mux mode ALT2.

This device does not have a dedicated OTG ID detect pin. For OTG ID pin detection, if needed, use one of the GPIO pins connected to the WUU configured as an input pin with pullup enabled.

61.3.3.3 LPUART over USB capability

This device supports a connection whereby any LPUART may connect to the FS USB physical layer (and the USB controller disconnected) to allow simple debugging capabilities for applications that do not have direct physical access to other LPUART Tx/Rx pins.

When you write 1 to USBCTRL[UARTSEL], the selected LPUART connects to the FS USB physical layer. This configures the FS USB0_DP/DM signals to configure as normal LPUART signals, which do not operate in differential mode. When USBCTRL[UARTCHLS] becomes 0, FS USB0_DP/DM signal pins are configured for LPUART Tx/Rx functions respectively. When USBCTRL[UARTCHLS] becomes 1, FS USB0_DP/DM signal pins are configured for LPUART Rx/Tx functions.

61.3.3.4 Lane reversal feature

This device supports lane reversal for the USB0_DP and USB0_DM pins for flexibility in PC board design depending on the USB receptacle type and location choices.

USBCTRL[DPDM_LANE_REVERSE] controls the lane reversal. After USBCTRL[DPDM_LANE_REVERSE] becomes 0, the USB0_DP and USB0_DM external package pins have their normal roles. After USBCTRL[DPDM_LANE_REVERSE] becomes 1, the I/O pin roles are switched so that the USB0_DP external package pin operates as USB_DM and the USB0_DM external package pin operates as USB_DP.

Lane reversal impacts all USB standard data communication functions along with battery charging detection and signaling. Lane reversal does **not** impact LPUART over USB mode selections for USB0_DP/DM pin functions because there is already a pin configuration for that mode using USBCTRL[UARTCHLS].

61.3.4 Modes of operation

USBFS includes the following main modes of operation:

- Active mode

- Low-Power mode
- Keep-Alive mode

61.3.4.1 Active mode

This is the basic mode of operation. The USBFS core uses software control to switch between two modes of operation:

- USB peripheral device
- USB host

In either mode, the USB host or USB peripheral device uses the same data paths and BDs for the transmission and reception of data. For this reason, use a USBFS core-centric nomenclature to describe the direction of the data transfer between the USBFS core and USB:

- Receive (or "Rx") describes transfers that move data from USB to memory.
- Transmit (or "Tx") describes transfers that move data from memory to USB.

The following table shows how the data direction corresponds to the USB token type in host and peripheral device applications.

Table 479. Data direction for USB host or USB peripheral device

	Rx	Tx
Device	OUT or SETUP	IN
Host	IN	OUT or SETUP

61.3.4.2 Low-Power mode

The USB controller supports Sleep mode, Deep Sleep mode, and Power Down mode to save power consumption.

When the USB subsystem detects no activity on the USB bus for more than 3 ms, ISTAT[SLEEP] becomes 1. This field can cause an interrupt, and you decide on the appropriate action.

Waking from a Sleep Low-Power mode when the USB subsystem is powered occurs through an asynchronous interrupt triggered by activity on the USB bus. Writing 1 to USBTRC0[USBRESMEN] enables this function.

The following wakeup features are also supported:

- In Active and Sleep modes, the USB controller can generate an interrupt on VREGIN detection using MISCCTRL[VFEDG_EN], MISCCTRL[VREDG_EN], USBTRC0[VFEDG_DET], and USBTRC0[VREDG_DET].
- In Deep Sleep and Power Down modes, VDD_LDO_USB is an input pin to WUU, and a transition on it can generate a wakeup.
- In Deep Sleep and Power Down modes, USB0_DP and USB0_DM are input pins to WUU, and a transition on those pins can generate a wake-up provided the USB subsystem is powered and in Host mode.
- In Deep Sleep and Power Down modes, the GPIO pins chosen to detect VBUS and OTG ID are also input to WUU, so transitions on them can generate a wakeup.

In Deep Power Down mode, the pads become isolated, and USB bus activity cannot wake the system.

61.3.4.3 Keep-Alive mode

Keep Alive mode is for the chips operating with Device-only USB use cases.

NOTE

The address range 48010000h to 480101FFh consists of 512 bytes of on-chip USB RAM, which is available in Active or Sleep modes. This allows the USB core to access the USB RAM in Keep Alive Mode.

When entering Keep Alive Mode, you must place the BDT entries and all USB buffers accessed during Keep Alive mode in the USB RAM space.

Keep Alive mode keeps USB basic response functionality active in the Sleep low-power mode, so there is no need to re-enumerate the USB device after exiting that Low-Power mode. When connected to the host and when there is no data to transfer, to save power, the software can power down the system bus and the CPU, keeping only the USB power on. In Keep Alive mode, only the USB 48 MHz functional clock is required by the USB controller; the rest of the device can transition to Sleep mode. When the SETUP token or a software-assigned token is received, the USB core wakes up, but it responds with NAK to all other types of packets.

Because the USB core must access the BDT in Keep Alive mode, use the on-chip USB RAM for the BDT and small data packets during Keep Alive mode. Place the BDT into this RAM in Keep Alive mode and changes all OWN fields to 0 (except the SETUP endpoint). You can do this by writing 1 to KEEP_ALIVE_CTRL[OWN_OVERRD_EN].

61.3.4.3.1 Entering Keep-Alive mode

To enter Keep-Alive mode:

1. **To configure the USB subsystem for Keep-Alive mode before entering Sleep mode:**
   - Initialize the BDT in USB RAM, including writing 0 to the OWN field within each BDT entry for every endpoint except for EP0. You must configure the buffer address for EP0 to an address within the USB SRAM, ensuring that you place received data within the USB SRAM. The USB FS subsystem cannot access other memory locations in USB Keep-Alive mode.
   - Configure KEEP_ALIVE_WKCTRL[WAKE_ON_THIS] appropriately to configure what received token generates the wakeup from Keep-Alive mode.
   - Enable the wakeup interrupt by writing 1 to KEEP_ALIVE_CTRL[WAKE_INT_EN].
   - Enable Keep-Alive mode operation by writing 1 to KEEP_ALIVE_CTRL[KEEP_ALIVE_EN].
2. **Enter Sleep mode, ensuring that the USB 48 MHz functional clock source remains enabled:**
   - If the 48 MHz clock is not configured for USB crystal-less operation, follow Device mode FIRC operation instructions.
   - The software then enters Sleep mode, and the USB subsystem is in Keep-Alive mode.
   - Follow the procedures described in the "Power Management" and "System Power Control (SPC)" chapters of this document to reduce power consumption in Sleep mode for extraneous modules.

61.3.4.3.2 Exiting Keep-Alive mode

Any enabled interrupt generates a wakeup from Sleep mode; this causes an exit from USB Keep Alive mode. You can also configure the USB to generate an interrupt when it receives a particular token (for example, SETUP), as configured by KEEP_ALIVE_WKCTRL[WAKE_ON_THIS]. Other USB interrupts can also remain enabled and are generated in USB Keep-Alive mode.

61.3.5 Clocking

This section describes clocks and special clocking requirements of USBFS.

Table 480. Clocking

Clock name	Description
Bus clock	The bus clock is system-dependent. It must be greater than 24 MHz.
Function clock	USB 48 MHz functional clock

61.3.6 Reset

Chip reset

The logic and registers for USBFS reset to their default states on a chip reset.

Software reset

USBFS implements a software reset field in its control register. See USBTRC0[USBRESET] for more information.

61.3.7 Interrupts

The following table illustrates the status flags that can generate USBFS interrupts.

Table 481. Interrupts

Flag	Description	Low-power wakeup
ONEMSEC	One millisecond timer timeout flag	No
LINE_STATE_CHG	Line state change flag	No
STALL	Stall handshake flag	No
ATTACH	Attach flag	No
RESUME	Resume flag	No
SLEEP	Sleep flag	No
TOKDNE	Token processing done flag	No
SOFTOK	SOF token flag	No
ERROR	Error flag	No
USBRST	USB reset flag	No
BTSERR	Bit stuff error flag	No
OWNERR	BD unavailable error flag	No
DMAERR	DMA access error flag	No
BTOERR	Bus turnaround timeout error flag	No
DFN8	Data field bit width error flag	No
CRC16	CRC16 error flag	No
CRC5EOF	EOF error flag	No
PIDERR	PID error flag	No
USB_RESUME_INT	Asynchronous resume wakeup flag	Yes
VFEDG_DET	VREGIN falling edge flag	Yes
VREDG_DET	VREGIN rising edge flag	Yes
WAKE_INT_STS	Wakeup interrupt flag in Keep-Alive mode	No
OVF_ERROR	Frequency trim adjustment flag	No

61.3.8 DMA

USBFS provides an integrated DMA controller that transfers the packet data to and from memory. The DMA controller allows USB endpoint data to transfer very efficiently, limiting processor involvement.

61.4 External signals

This section describes clocks and special clocking requirements of USBFS.

Table 482. External signals

Signal	Description
USB0_DP	USB D+ pin
USB0_DM	USB D- pin
VDD_USB	3.3 V power supply for USB transceiver

61.5 Initialization

For Host mode initialization, see Host mode operation examples.

For Device mode initialization, see Device mode operation examples.

61.6 Application information

61.6.1 OTG and Host mode operation

The Host mode logic allows portable, mobile, and other devices using this chip to function as a USB-embedded host (EH) controller. The OTG logic adds an interface to allow the OTG host negotiation and session request protocols (HNP and SRP) to implement in software.

Host mode is used in handheld-portable devices to connect simple HID class devices like printers and keyboards easily. It is not intended to perform the functions of a full OHCI or UHCI-compatible host controller found on PC motherboards. Host mode allows bulk, isochronous, interrupt, and control transfers. Perform bulk data transfers at nearly the total USB interface bandwidth. Support is provided for ISO transfers, but the number of ISO streams you can practically support is affected by the interrupt latency of the processor servicing the Token Done interrupts from the SIE. You must write custom drivers to support host mode operation.

The USBFS core can only operate as a peripheral device or in Host mode; the USBFS core cannot operate simultaneously in both peripheral device and Host modes. Writing 1 to CTL[HOSTMODEEN] enables Host mode. When Host mode is enabled, use the endpoint zero only. You must disable all other endpoints.

61.6.2 Host mode operation examples

The following sections list the steps to perform USB host functions using the USBFS core. See the *Universal Serial Bus Specification, Revision 2.0*, "Chapter 9 USB Device Framework" for more information.

To enable Host mode and discover a connected device:

1. Enable Host Mode (CTL[HOSTMODEEN] = 1).
   - Enable the pulldown resistors and disable the pullup resistors.
   - SOF generation begins.
   - USBFS initializes the SOF counter to 12,000 (corresponding to an interval of 1 ms between SOF frames).
   - To eliminate noise on the USB, write 0 to CTL[USBENSOFEN] to disable SOF packet generation.
2. Enable the ATTACH interrupt (INTEN[ATTACHEN] = 1).
3. Wait for the ATTACH interrupt (ISTAT[ATTACH]), which is signaled by a USB peripheral device pullup resistor changing the state of DPLUS or DMINUS from 0 to 1 (CTL[SE0] to J or K state).
4. Check the state of CTL[JSTATE] and CTL[SE0]. If the connecting device is low speed (CTL[JSTATE] is 0), then:
   - Write 1 to ADDR[LSEN].
   - Write 1 to ENDPT0[HOSTWOHUB].
5. Enable RESET (CTL[RESET] = 1) for 10 ms.
6. Enable SOF packet to keep the connected device from going suspended (CTL[USBENSOFEN] = 1).

7. Send the appropriate commands to the default control pipe of the connected device, to enumerate the attached device.

To complete a control transaction to a connected device:

1. Complete all the steps to discover a connected device.
2. Set up the endpoint control register for bidirectional control transfers: ENDPT0[4:0] = 0dh.
3. Place a copy of the device framework setup command in a memory buffer.
4. Initialize current even or odd TX EP0 BDT to transfer the 8 bytes of command data for a device framework command (for example, get a device descriptor).
   - Write 0008_0080h to the BDT command word –Byte count to 8, and write 1 to OWN.
   - Set the BDT buffer address field (see Table 476) buffer address field to the start address of the 8-byte command buffer.
5. Set the USB device address of the peripheral device in the address register (ADDR[ADDR]). After the USB bus reset, the chip's USB address is zero. The set address device framework command writes 1 to the chip USB address.
6. Write Token (TOKEN) with a SETUP to Endpoint 0, the peripheral device default control pipe (TOKEN = D0h).
   - This initiates a setup token on the bus, followed by a data packet.
   - After completing the packets, return the device handshake in the BDT PID field.
   - When you write the BDT, assert a Token Done (ISTAT[TOKDNE]) interrupt. This completes the setup phase of the setup transaction.
7. To initiate the data phase of the setup transaction (that is, to get the data for the GET DEVICE DESCRIPTOR command), set up a buffer in memory for the data to be transferred.
8. Initialize the current even or odd TX EP0 BDT to transfer the data.
   - Write 0040_00C0h to the BDT command word – BC to 64 (the byte count of the data buffer, in this case), write 1 to OWN field, and set Data toggle to Data1.
   - Set the BDT buffer address field to the start address of the data buffer.
9. Write Token (TOKEN) with an IN or OUT token to Endpoint 0 (the peripheral device default control pipe), and write an IN token for a GET DEVICE DESCRIPTOR command (TOKEN = 0x90). This initiates an IN token on the bus, followed by a data packet from the device to the host.

   After the data packet completes, write the BDT and assert a Token Done interrupt (ISTAT[TOKDNE]). For control transfers with a single packet data phase, this completes the data phase of the setup transaction.
10. To initiate the status phase of the setup transaction, set up a buffer in memory to receive or send the zero length status phase data packet.
11. Initialize the current even or odd TX EP0 BDT to transfer the status data.
    - Write 0000_00C0h to the BDT command word – BC to 0 (the byte count of the data buffer in this case), write 1 to the OWN field, and set the Data toggle to Data1.
    - Set the BDT buffer address field to the start address of the data buffer.
12. Write Token (TOKEN) with an IN or OUT token to Endpoint 0 (the peripheral device default control pipe), and write an OUT token for a GET DEVICE DESCRIPTOR command (TOKEN = 10h). This initiates an OUT token on the bus, followed by a *zero-length* data packet from the host to the device.

    After the data packet completes, write the BDT with the handshake from the device, and assert a Token Done (ISTAT[TOKDNE]) interrupt. This completes the data phase of the setup transaction.

To send a full speed bulk data transfer to a peripheral device:

1. Complete all steps to discover and configure a connected device. Write Address (ADDR) with the address of the peripheral device. Typically, there is only one other device on the USB bus in Host mode, so it is expected that the address is 01h and must remain constant.
2. Write 1Dh to Endpoint Control (ENDPT0 - ENDPT15) to enable transmit and receive transfers with handshaking enabled.
3. Set up the even TX EP0 BDT to transfer up to 64 bytes.
4. Set the USB device address of the peripheral device in the address register (ADDR[ADDR]).
5. Write Token (TOKEN) with an OUT token to the desired endpoint. The write to Token (TOKEN) triggers the USBFS transmit state machines to begin transmitting the token and the data.
6. Set up the odd TX EP0 BDT to transfer up to 64 bytes.
7. Write Token (TOKEN) with an OUT token, as in 5. To achieve maximum throughput, you can queue two tokens simultaneously to allow the packets to be double-buffered.
8. Wait for ISTAT[TOKDNE] interrupt. This indicates that one of the BDTs has been released back to the processor and that the transfer has completed.
   - If the peripheral device asserts NAKs, then USBFS continues to retry the transfer indefinitely without processor intervention (unless ENDPT0[RETRYDIS] is 1).
   - If ENDPT0[RETRYDIS] becomes 1, then return the handshake (ACK, NAK, STALL, or ERROR (fh)) in the BDT PID field.
   - If a stall interrupt occurs, then you must de-queue the pending packet and you must clear the error condition in the peripheral device.
   - If a Reset interrupt occurs (CTL[SE0] for more than 2.5 µs), the peripheral device detaches.
9. After ISTAT[TOKDNE] interrupt occurs, return to step 2 to examine the BDTs and queue the next data packet.

61.6.3 Device mode operation examples

The following sections list the steps required to perform USB device functions using the USBFS core.

To enable Device mode and discover a connected USB host:

1. Disable Host Mode (CTL[HOSTMODEEN] = 0).
2. Pullup DP (CONTROL[DPPULLUPNONOTG] = 1b).
3. Disable weak pulldown on DP and DM (USBCTRL[PDE] = 0b).
4. Disable suspend (USBCTRL[SUSP] = 1'b0).
5. Clear status flags.
6. Enable USBRST interrupt (INTEN[USBRSTEN] = 1b).
7. Enable USBFS (CTL[USBENSOFEN] = 1'b1).
8. Wait for reset request from USB host (ISTAT[USBRST] = 1b).
9. Detect USBRST interrupt, device connected to USB host.

To complete a packet transfer:

1. Complete all the steps to discover a connected USB host.
2. Set up the endpoint control register for packet transfer (ENDPT0[4:0] = 0dh).
3. Enable TOKDNE interrupt (INTEN[TOKDNEEN] = 1b).
4. Initialize the endpoint BDT in USB RAM including setting OWN = 1b.
5. Clear TXD_SUSPEND flag (write 1 to clear CTL[TXSUSPENDTOKENBUSY]).

6. Wait packet transfer to complete (ISTAT[TOKDNE] = 1b).
7. Detect TOKDNE interrupt, check BDT and status flags.

61.6.4 OTG operation

The USB-OTG core provides sensors and controls to enable On-The-Go (OTG) operation. The OTG driver software uses these sensors to implement the host negotiation protocol (HNP) and session request protocol (SRP) and to give access to the OTG protocol control signals. The following state machines show the OTG operations involved with HNP and SRP protocols from either end of the USB cable.

61.6.4.1 OTG dual role A device operation

A device is considered the A device:

- According to the type of cable attached.
- If you plug in the USB Type Standard-A or Micro-A plug into the device.

The following flow diagram and state description table show how a *dual role* A device operates:

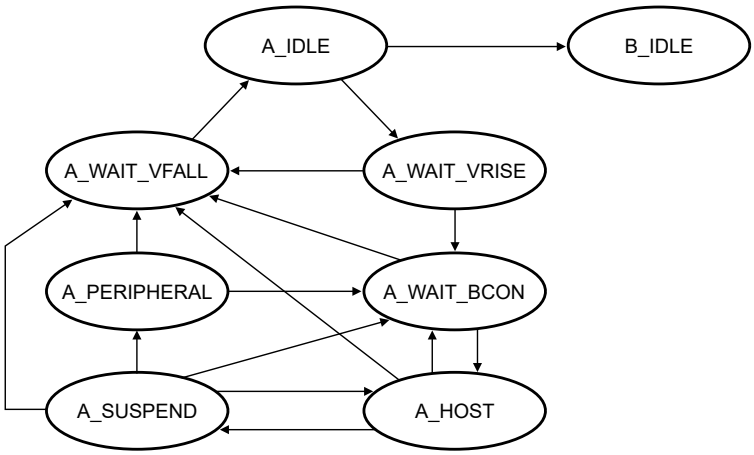


Figure 336. Dual role A device flow diagram

Table 483. State descriptions for the dual role A device flow

State	Possible events in this state	Event response
A_IDLE	An ID interrupt[1] occurs, indicating that the cable has been unplugged, or a Type B cable is attached. The device now acts as a Type B device.	Go to B_IDLE.
	The A application wants to use the bus or the B device is doing an SRP (as A_SESS_VLD interrupt or Attach or Port status change interrupt indicates).	Go to A_WAIT_VRISE. Turn on DRV_VBUS. Check the data line for 5-10 ms pulsing.
A_WAIT_VRISE	An ID interrupt[1] occurs or if A_VBUS_VLD is false after 100 msec, indicating that the cable is changed or the A device cannot support the current required from the B device.	Go to A_WAIT_VFALL. Turn off DRV_VBUS.
	An A_VBUS_VLD interrupt occurs	Go to A_WAIT_BCON.

Table continues on the next page...

Table 483. State descriptions for the dual role A device flow (continued)

State	Possible events in this state	Event response
A_WAIT_BCON	200 ms pass without an Attach or ID interrupt[1] (this could wait forever if desired)	Go to A_WAIT_FALL. Turn off DRV_VBUS.
	An A_VBUS_VLD interrupt occurs and a B device attaches	Go to A_HOST. Enter Host mode.
A_HOST	Enumerate the device to determine OTG support. Then one of the following occurs:	
	• An A_VBUS_VLD interrupt occurs. • A device has finished its session and does not need the bus anytime soon. • The B device disconnects.	Go to A_WAIT_VFALL. Exit Host mode. Turn off DRV_VBUS.
	One of the following occurs: • The A device finishes its session. • The A device relinquishes the bus to the B device.	Go to A_SUSPEND.
	An ID interrupt[1] occurs or if the B device disconnects.	Go to A_WAIT_BCON.
A_SUSPEND	An ID interrupt[1] occurs, or if a 150 ms B disconnect timeout occurs (this timeout value could be longer), or if an A_VBUS_VLD interrupt occurs.	Go to A_WAIT_VFALL. Turn off DRV_VBUS.
	HNP is enabled, and B disconnects in 150 ms, then B device becomes the host.	Go to A_PERIPHERAL. Exit Host mode.
	The A device wants to start another session.	Go to A_HOST.
A_PERIPHERAL	An ID interrupt[1] or A_VBUS_VLD interrupt occurs.	Go to A_WAIT_VFALL. Turn off DRV_VBUS.
	The bus is idle for 3–200 ms.	Go to A_WAIT_BCON. Enter Host mode.
A_WAIT_VFALL	An ID interrupt[1] occurs or (A_SESS_VLD or a B device connects)	Go to A_IDLE.

1. ID interrupt is triggered due to a detected change in the resistance between the local USB connector ID pin and ground.

61.6.4.2 OTG dual role B device operation

A device is considered a B device if you connect it to the bus with a USB type Standard-B, Mini-B, or Micro-B plug inserted into the local USB receptacle. The type Mini-B plug and receptacle are now only allowed for dedicated peripheral devices and not allowed for a dual role or OTG devices.

The following flow diagram and state description table show how a *dual role* B device operates:

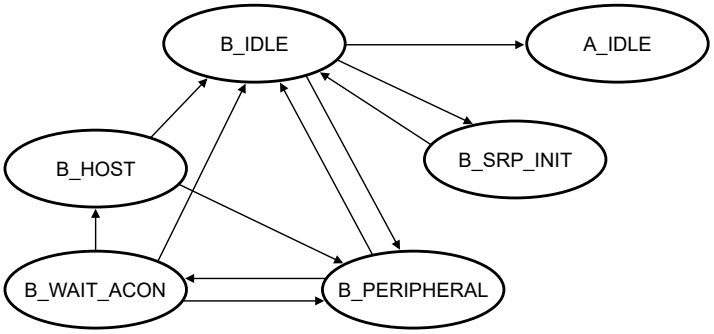


Figure 337. Dual role B device flow diagram

Table 484. State descriptions for the dual role B device flow

State	Possible events in this state	Event response
B_IDLE	An ID interrupt[1] occurs, indicating that a Type A cable is plugged in. The device must now respond as a type A device.	Go to A_IDLE.
	A B_SESS_VLD interrupt occurs, and the A device has turned on VBUS and begins a session.	Go to B_PERIPHERAL. Turn on DP_HIGH.
	A B application wants the bus and the bus is idle for 2 ms, and B_SESS_END becomes 1, then the B device can perform an SRP.	Go to B_SRP_INIT. Pulse CHRG_VBUS Pulse. DP_HIGH 5-10 ms.
B_SRP_INIT	One of the following occurs: • An ID interrupt[1] • The B device completes initiating SRP	Go to B_IDLE.
B_PERIPHERAL	HNP is enabled and the bus is suspended, and the B device wants the bus, B can become the host.	Go to B_WAIT_ACON. Turn off DP_HIGH.
B_WAIT_ACON	An A device connects, and an attach interrupt is received.	Go to B_HOST. Exit Host Mode.
	An ID interrupt[1] or B_SESS_VLD interrupt occurs. The cable changes or VBUS goes away, the host does not support it.	Go to B_IDLE.
	3.125 ms expires or a Resume occurs	Go to B_PERIPHERAL.
B_HOST	An ID interrupt[1] or B_SESS_VLD interrupt occurs. The cable changes or if VBUS goes away, then the host does not support it.	Go to B_IDLE.
	B application is done or the A device disconnects.	Go to B_PERIPHERAL.

1. An ID interrupt is triggered due to a detected change in the resistance between the local USB connector ID pin and ground.

61.7 Memory map and register definitions

This section provides the memory map and detailed descriptions of all USBFS interface registers.

NOTE

OTG Control (OTGCTL), Control (CTL), USB Control (USBCTRL), and USB OTG Control (CONTROL) have similar names but these are all separate registers.

61.7.1 USBFS memory mapping

In addition to the address space occupied by the USBFS registers, 512 bytes of embedded RAM are required for use in Keep Alive mode. On this chip the starting address of the USB RAM block is 4801_0000h.

You can use this USB RAM in Active power mode for BDT entries, USB data buffers, or other purposes.

61.7.2 USBFS register descriptions

61.7.2.1 USBFS memory map

USBFS0 base address: 400D_D000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Peripheral ID (PERID)	8	R	04h
4h	Peripheral ID Complement (IDCOMP)	8	R	FBh
8h	Peripheral Revision (REV)	8	R	33h
Ch	Peripheral Additional Information (ADDINFO)	8	R	01h
10h	OTG Interrupt Status (OTGISTAT)	8	RW	00h
14h	OTG Interrupt Control (OTGICR)	8	RW	00h
18h	OTG Status (OTGSTAT)	8	R	00h
1Ch	OTG Control (OTGCTL)	8	RW	00h
80h	Interrupt Status (ISTAT)	8	RW	00h
84h	Interrupt Enable (INTEN)	8	RW	00h
88h	Error Interrupt Status (ERRSTAT)	8	RW	00h
8Ch	Error Interrupt Enable (ERREN)	8	RW	00h
90h	Status (STAT)	8	R	00h
94h	Control (CTL)	8	RW	00h
98h	Address (ADDR)	8	RW	00h
9Ch	BDT Page 1 (BDTPAGE1)	8	RW	00h
A0h	Frame Number Register Low (FRMNUML)	8	R	00h
A4h	Frame Number Register High (FRMNUMH)	8	R	00h
A8h	Token (TOKEN)	8	RW	00h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
ACh	SOF Threshold (SOFTHLD)	8	RW	00h
B0h	BDT Page 2 (BDTPAGE2)	8	RW	00h
B4h	BDT Page 3 (BDTPAGE3)	8	RW	00h
C0h	Endpoint Control (ENDPT0)	8	RW	00h
C4h	Endpoint Control (ENDPT1)	8	RW	00h
C8h	Endpoint Control (ENDPT2)	8	RW	00h
CCh	Endpoint Control (ENDPT3)	8	RW	00h
D0h	Endpoint Control (ENDPT4)	8	RW	00h
D4h	Endpoint Control (ENDPT5)	8	RW	00h
D8h	Endpoint Control (ENDPT6)	8	RW	00h
DCh	Endpoint Control (ENDPT7)	8	RW	00h
E0h	Endpoint Control (ENDPT8)	8	RW	00h
E4h	Endpoint Control (ENDPT9)	8	RW	00h
E8h	Endpoint Control (ENDPT10)	8	RW	00h
ECh	Endpoint Control (ENDPT11)	8	RW	00h
F0h	Endpoint Control (ENDPT12)	8	RW	00h
F4h	Endpoint Control (ENDPT13)	8	RW	00h
F8h	Endpoint Control (ENDPT14)	8	RW	00h
FCh	Endpoint Control (ENDPT15)	8	RW	00h
100h	USB Control (USBCTRL)	8	RW	C0h
104h	USB OTG Observe (OBSERVE)	8	R	50h
108h	USB OTG Control (CONTROL)	8	RW	00h
10Ch	USB Transceiver Control 0 (USBTRC0)	8	RW	00h
114h	Frame Adjust (USBFRMADJUST)	8	RW	00h
124h	Keep Alive Mode Control (KEEP_ALIVE_CTRL)	8	RW	08h
128h	Keep Alive Mode Wakeup Control (KEEP_ALIVE_WKCTRL)	8	RW	01h
12Ch	Miscellaneous Control (MISCCTRL)	8	RW	00h
130h	Peripheral Mode Stall Disable for Endpoints 7 to 0 in IN Direction (STALL_IL_DIS)	8	RW	00h
134h	Peripheral Mode Stall Disable for Endpoints 15 to 8 in IN Direction (STALL_IH_DIS)	8	RW	00h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
138h	Peripheral Mode Stall Disable for Endpoints 7 to 0 in OUT Direction (STALL_OL_DIS)	8	RW	00h
13Ch	Peripheral Mode Stall Disable for Endpoints 15 to 8 in OUT Direction (STALL_OH_DIS)	8	RW	00h
140h	USB Clock Recovery Control (CLK_RECOVER_CTRL)	8	RW	00h
144h	FIRC Oscillator Enable (CLK_RECOVER_IRC_EN)	8	RW	01h
154h	Clock Recovery Combined Interrupt Enable (CLK_RECOVER_INT_EN)	8	RW	10h
15Ch	Clock Recovery Separated Interrupt Status (CLK_RECOVER_INT_STATUS)	8	RW	00h

61.7.2.2 Peripheral ID (PERID)

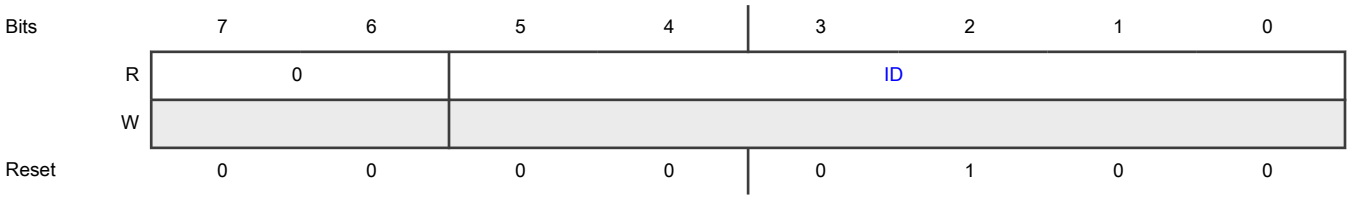
Offset

Register	Offset
PERID	0h

Function

Reads back the value of 04h. This value is defined for the USB peripheral.

Diagram



Fields

Field	Function
7-6 —	Reserved
5-0 ID	Peripheral Identification

61.7.2.3 Peripheral ID Complement (IDCOMP)

Offset

Register	Offset
IDCOMP	4h

Function
Indicates the one's complement of Peripheral ID (PERID). For the USB peripheral, the value is FBh.

Diagram

Bits	7	6	5	4	3	2	1	0
R	1				NID			
W								
Reset	1	1	1	1	1	0	1	1

Fields

Field	Function
7-6 —	Reserved
5-0 NID	Negative Peripheral ID Indicates the one's complement of PERID[ID].

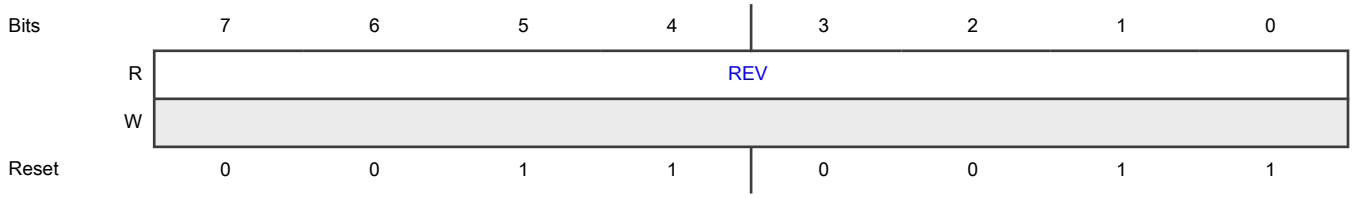
61.7.2.4 Peripheral Revision (REV)

Offset

Register	Offset
REV	8h

Function
Indicates the revision number of USBFS.

Diagram



Fields

Field	Function
7-0 REV	Revision Contains the USBFS revision number.

61.7.2.5 Peripheral Additional Information (ADDINFO)

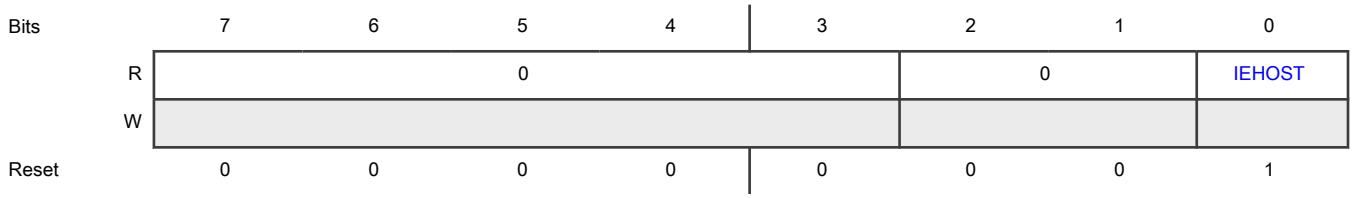
Offset

Register	Offset
ADDINFO	Ch

Function

Indicates whether Host mode is enabled.

Diagram



Fields

Field	Function
7-3 —	Reserved
2-1 —	Reserved
0 IEHOST	Host Mode Enable 0b - Disabled 1b - Enabled

61.7.2.6 OTG Interrupt Status (OTGISTAT)

Offset

Register	Offset
OTGISTAT	10h

Function

Records changes to the state of the 1-millisecond timer and line state stable logic.

You can read this register to determine the event that triggers an interrupt.

Only fields that have changed since the last read becomes 1.

Diagram

Bits	7	6	5	4	3	2	1	0
R	Reserved	ONEMSEC	LINE_ST...	0	Reserved	Reserved	0	Reserved
W	W1C	W1C	W1C		W1C	W1C		W1C
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 —	Reserved
6 ONEMSEC	One Millisecond Timer Timeout Flag Specifies the status of the 1 millisecond timer. This field becomes 1 when the 1 millisecond timer expires. To avoid losing 1 ms counts, service the interrupt every millisecond. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not timed out 1b - Timed out When writing 0b - No effect 1b - Clear the flag
5	Line State Change Interrupt Flag Detects Reset, Resume, Connect, and Data Line Pulse signaling.

Table continues on the next page...

Table continued from the previous page...

Field	Function
LINE_STATE_CHG	• This field becomes 1 when the USB line state (CTL [SE0] and CTL [JSTATE]) are stable without change for 1 ms, and the value of the line state varies from the last time when the line state was stable. • This field is 1 on transitions between CTL[SE0] and CTL[JSTATE], CTL[SE0] and K-state, and CTL[JSTATE] and K-state. Changes in CTL[JSTATE] when CTL[SE0] is true and do not cause an interrupt. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
4 —	Reserved
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

61.7.2.7 OTG Interrupt Control (OTGICR)

Offset

Register	Offset
OTGICR	14h

Function
Enables the corresponding interrupt status fields defined in OTG Interrupt Status (OTGISTAT).

Diagram

Bits	7	6	5	4	3	2	1	0
R W	Reserved	ONEMSEC...	LINESTA...	Reserved	Reserved	Reserved	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 —	Reserved
6 ONEMSECEN	1-Millisecond Interrupt Enable Enables the 1 ms timer interrupt. 0b - Disable 1b - Enable
5 LINESTATEEN	Line State Change Interrupt Enable Enables the line state change interrupt. 0b - Disable 1b - Enable
4 —	Reserved
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

61.7.2.8 OTG Status (OTGSTAT)

Offset

Register	Offset
OTGSTAT	18h

Function

Displays the actual value from the 1 millisecond timer and line state stable logic.

Diagram

Bits	7	6	5	4	3	2	1	0
R	0	ONEMSEC	LINESTA...	0				
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 —	Reserved
6 ONEMSEC	Reserved for 1 ms count
5 LINESTATESTABLE	Line State Stable Indicates that the internal signals that control OTGISTAT[LINE_STATE_CHG] are stable for at least 1 ms. This field provides hardware debounce of the linestate in detecting connect, disconnect, and resume signaling. First read OTGISTAT[LINE_STATE_CHG] and then read this field. If OTGISTAT[LINE_STATE_CHG] reads as 1, you can consider the value of OTGISTAT[LINE_STATE_CHG] stable. 0b - Unstable 1b - Stable
4-0 —	Reserved

61.7.2.9 OTG Control (OTGCTL)

Offset

Register	Offset
OTGCTL	1Ch

Function

Controls the operation of data line termination resistors.

Diagram

Bits	7	6	5	4	3	2	1	0
R W	DPHIGH	Reserved	DPLOW	DMLOW	Reserved	OTGEN	Reserved	
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 DPHIGH	D+ Data Line Pullup Resistor Enable 0b - Disable 1b - Enable
6 —	Reserved
5 DPLOW	D+ Data Line pulldown Resistor Enable If you write 1 to this field, ensure that you also write 1 to OTGCTL[DMLOW]. 0b - Disable 1b - Enable
4 DMLOW	D– Data Line Pulldown Resistor Enable 0b - Disable 1b - Enable
3 —	Reserved
2 OTGEN	On-The-Go Pullup and Pulldown Resistor Enable Enables OTG pullup and pulldown resistor. 0b - If USBENSOFEN is 1 and HOSTMODEEN is 0 in the Control Register (CTL), then the D+ Data line pullup resistors are enabled. If HOSTMODEEN is 1, then the D+ and D– Data line pulldown resistors are engaged. 1b - Uses the pullup and pulldown controls in this register.
1-0 —	Reserved

61.7.2.10 Interrupt Status (ISTAT)

Offset

Register	Offset
ISTAT	80h

Function

Contains flag fields for each of the interrupt sources within USBFS. Each of these fields is qualified with their respective interrupt enable fields.

All fields of this register are logically OR'd together, along with OTG Status (OTGSTAT), to form a single interrupt source for the processor's interrupt controller.

Diagram

Bits	7	6	5	4	3	2	1	0
R	STALL	ATTACH	RESUME	SLEEP	TOKDNE	SOFTOK	ERROR	USBRST
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 STALL	Stall Interrupt Flag In Device mode, becomes 1 when SIE sent a STALL handshake. In Host mode, USBFS changes this field to 1 when USBFS detects a STALL acknowledge during the handshake phase of a USB transaction. This interrupt can determine whether the last USB transaction was completed successfully or stalled. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
6 ATTACH	Attach Interrupt Flag Indicates when USBFS detects an attach of a USB device. This field is only meaningful if CTL[HOSTMODEEN] = 1. This field signifies that a peripheral is now present and must be configured. Asserts if no transitions occur on the USB for 2.5 µs and the current bus state is not CTL[SE0].

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
5 RESUME	Resume Flag Indicates when a K-state is observed on the DP or DM signals for 2.5 µs. You must disable this interrupt when not in Suspend mode. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
4 SLEEP	Sleep Flag Indicates when USBFS detects a constant idle on the USB bus for 3 ms. Activity on the USB bus resets the sleep timer. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
3 TOKDNE	Current Token Processing Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates when the current token being processed has completed. The processor must immediately read Status (STAT) to determine the EndPoint and BD used for this token. Clearing this flag causes one of the following events: • STAT becomes 0. • USBFS loads the STAT holding register into Status (STAT). **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not processed 1b - Processed When writing 0b - No effect 1b - Clear the flag
2 SOFTOK	Start Of Frame (SOF) Token Flag Indicates when USBFS receives a SOF token. In Host mode, this field becomes 1 when either of the following events occur. After that, you can prepare for the next SOF. • The SOF threshold is reached (MISCCTRL[SOFBUSSET] = 0). • The SOF counter reaches 0 (MISCCTRL[SOFBUSSET] = 1). **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not receive 1b - Received When writing 0b - No effect 1b - Clear the flag
1 ERROR	Error Flag Indicates when any of the error conditions within Error Interrupt Status (ERRSTAT) occur. The processor must then read Error Interrupt Status (ERRSTAT) to determine the source of the error. **NOTE** This field behaves differently for register reads and writes. When reading

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Error did not occur 1b - Error occurred When writing 0b - No effect 1b - Clear the flag
0 USBRST	USB Reset Flag Indicates when USBFS detects a valid USB reset. This informs the processor that it must write 00h into the address register and enable endpoint 0. This field becomes 1 after a USB reset is detected for 2.5 µs. This field does not become 1 again until the USB reset condition is removed and the USB reset condition is reasserted. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag

61.7.2.11 Interrupt Enable (INTEN)

Offset

Register	Offset
INTEN	84h

Function

Enables interrupt sources within USBFS. Writing 1 to any of these fields enables the respective interrupt source in Interrupt Status (ISTAT).

Diagram

Bits	7	6	5	4	3	2	1	0
R W	STALLEN	ATTACHEN	RESUMEEN	SLEEPEN	TOKDNEEN	SOFTOKEN	ERROREN	USBRSTEN
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 STALLEN	STALL Interrupt Enable Enables the STALL interrupt. 0b - Disable 1b - Enable
6 ATTACHEN	ATTACH Interrupt Enable Enables the ATTACH interrupt. 0b - Disable 1b - Enable
5 RESUMEEN	RESUME Interrupt Enable Enables the RESUME interrupt. 0b - Disable 1b - Enable
4 SLEEPEN	SLEEP Interrupt Enable Enables the SLEEP interrupt. 0b - Disable 1b - Enable
3 TOKDNEEN	TOKDNE Interrupt Enable Enables the TOKDNE interrupt. 0b - Disable 1b - Enable
2 SOFTOKEN	SOFTOK Interrupt Enable Enables the SOFTOK interrupt. 0b - Disable 1b - Enable
1 ERROREN	ERROR Interrupt Enable Enables the ERROR interrupt.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
0 USBRSTEN	USBRST Interrupt Enable Enables the USBRST interrupt. 0b - Disable 1b - Enable

61.7.2.12 Error Interrupt Status (ERRSTAT)

Offset

Register	Offset
ERRSTAT	88h

Function

Enables interrupt error sources within USBFS. Each of these fields are qualified with their respective error enable fields.

All fields of this register are logically OR'ed together and the result placed in ISTAT[ERROR].

Each field becomes 1 as soon as you detect the error condition; therefore, the interrupt does not typically correspond with the end of a token being processed.

Diagram

Bits	7	6	5	4	3	2	1	0
R	BTSERR	OWNERR	DMAERR	BTOERR	DFN8	CRC16	CRC5EOF	PIDERR
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 BTSERR	Bit Stuff Error Flag Indicates when a bit stuff error is detected. If this field is 1, then the corresponding packet is rejected due to the error. **NOTE** This field behaves differently for register reads and writes. When reading

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Packet not rejected due to the error 1b - Packet rejected due to the error When writing 0b - No effect 1b - Clear the flag
6 OWNERR	BD Unavailable Error Flag Indicates when USBFS operates in Peripheral mode (CTL[HOSTMODEEN] = 0). This field becomes 1 if USBFS requires a new BD for SETUP, ISO IN, or ISO OUT transfer when a new BD is not available. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
5 DMAERR	DMA Access Error Flag Indicates if USBFS has requested a DMA access to read a new BDT, but has not been given the bus before it needs to receive or transmit data. • If processing a transit transfer, this causes a transmit data underflow condition. • If processing a receive transfer, this causes a receive data overflow condition. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
4 BTOERR	Bus Turnaround Timeout Error Flag Indicates when a bus turnaround timeout error occurs. USBFS contains a bus turnaround timer that keeps track of the amount of time elapsed between the token and data phases of a SETUP or OUT TOKEN or the data and handshake phases of a IN TOKEN. A bus turnaround timeout error occurs if more than 16 bit times are counted from the previous EOP before a transition from IDLE.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - Not timed out 1b - Timed out When writing 0b - No effect 1b - Clear the flag
3 DFN8	Data Field Not 8 Bits Flag Indicates if the data field received was not 8 bits in length. USB Specification 1.0 requires that data fields be an integer number of bytes. If the data field is not an integer number of bytes, then this field becomes 1. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Integer number of bytes 1b - Not an integer number of bytes When writing 0b - No effect 1b - Clear the flag
2 CRC16	CRC16 Error Flag Indicates when a data packet is rejected due to a CRC16 error. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not rejected 1b - Rejected When writing 0b - No effect 1b - Clear the flag
1 CRC5EOF	CRC5 Error or End of Frame Error Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When USBFS operates in Peripheral mode (CTL[HOSTMODEEN] = 0), this interrupt detects CRC5 errors in the token packets generated by the host. If this field is 1, then the token packet is rejected because of a CRC5 error. When USBFS operates in Host mode (CTL[HOSTMODEEN] = 1), this interrupt detects End Of Frame (EOF) error conditions. This type of error occurs when USBFS transmits or receives data and the SOF counter reaches zero. This interrupt is useful when developing USB packet scheduling software to ensure that no USB transactions cross the start of the next frame. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
0 PIDERR	PID Error Flag Indicates when the PID check field fails. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not fail 1b - Failed When writing 0b - No effect 1b - Clear the flag

61.7.2.13 Error Interrupt Enable (ERREN)

Offset

Register	Offset
ERREN	8Ch

Function
Enables the error interrupt sources within USBFS.

Writing 1 to any of these fields enables the respective interrupt source in ERRSTAT.

Each field becomes 1 as soon as the error condition is detected; therefore, the interrupt does not typically correspond with the end of a token processed.

Diagram

Bits	7	6	5	4	3	2	1	0
R W	BTSERREN	OWNERREN	DMAERREN	BTOERREN	DFN8EN	CRC16EN	CRC5EOF...	PIDERREN
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 BTSERREN	BTSERR (Bit Stuff Error) Interrupt Enable 0b - Disable 1b - Enable
6 OWNERREN	OWNERR Interrupt Enable This field is valid when USBFS operates in Peripheral mode (CTL[HOSTMODEEN] = 0). 0b - Disable 1b - Enable
5 DMAERREN	DMAERR Interrupt Enable 0b - Disable 1b - Enable
4 BTOERREN	BTOERR (Bus Timeout Error) Interrupt Enable 0b - Disable 1b - Enable
3 DFN8EN	DFN8 (Data Field Not Integer Number of Bytes) Interrupt Enable 0b - Disable 1b - Enable
2 CRC16EN	CRC16 Interrupt Enable 0b - Disable 1b - Enable
1 CRC5EOFEN	CRC5/EOF Interrupt Enable 0b - Disable 1b - Enable
0 PIDERREN	PIDERR Interrupt Enable 0b - Disable 1b - Enable

61.7.2.14 Status (STAT)

Offset

Register	Offset
STAT	90h

Function

Reports the transaction status within USBFS.

When the processor's interrupt controller receives ISTAT[TOKDNE], you must read Interrupt Status (ISTAT) to determine the status of the previous endpoint communication. The data in this register is valid when the TOKDNE interrupt is asserted. This register is a read window into a group FIFO that USBFS maintains. After USBFS uses a BD, it updates this register.

If you perform another USB transaction before servicing the TOKDNE interrupt, then USBFS stores the status of the next transaction in the STAT FIFO. Thus, STAT is a 4-byte FIFO that allows the processor core to process one transaction when the SIE processes the next transaction. Writing 0 to ISTAT[TOKDNE] causes the SIE to update this register with the contents of the next STAT value. If the data in the STAT holding register is valid, then the SIE immediately reasserts to TOKDNE interrupt.

Diagram

Bits	7	6	5	4	3	2	1	0
R	ENDP				TX	ODD	0	
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7-4 ENDP	Endpoint address Encodes the endpoint address that receives or transmits the previous token. This allows the processor core to determine the BDT entry that the last USB transaction update.
3 TX	Transmit Indicator Indicates whether the most recent transaction was a receive or transmit operation. 0b - Receive 1b - Transmit
2 ODD	Odd Bank Indicates whether the last BD that you updated is in the odd bank of the BDT. 0b - Not in the odd bank 1b - In the odd bank
1-0 —	Reserved

61.7.2.15 Control (CTL)

Offset

Register	Offset
CTL	94h

Function
Provides various control and configuration information for USBFS.

Diagram

Bits	7	6	5	4	3	2	1	0
R	JSTATE	SE0	TXSUSPE...	RESET	HOSTMOD...	RESUME	ODDRST	USBENSO...
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 JSTATE	Live USB Differential Receiver JSTATE Signal Specifies that the current state of ADDR[LSEN] affects the polarity of this signal.
6 SE0	Live USB Single-Ended Zero signal
5 TXSUSPENDTOKENBUSY	TXD Suspend And Token Busy In Device mode, disables packet transmission and reception. SIE writes 1 to this field to disable this functionality. SIE does this when a SETUP token is received. This allows you to dequeue any pending packet transactions in the BDT before resuming token processing. Write 0 to this field to allow SIE to continue. In Host mode, USBFS writes 1 to this field when USBFS is busy executing a USB token. When this field is 1, you must not write more token commands to Token (TOKEN). Before writing any tokens to TOKEN, wait until this field becomes 0 to ensure that token commands are not lost.
4 RESET	Reset Signaling Enable Enables USBFS to generate USB reset signaling for resetting USB peripherals. This field is meaningful only in Host mode (CTL[HOSTMODEEN] = 1). You must write 1 to this field for the duration of reset signaling, then write 0 to this field to end reset signaling. 0b - Disable 1b - Enable
3 HOSTMODEEN	Host Mode Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - USBFS operates in Device mode. 1b - USBFS operates in Host mode. In Host mode, USBFS performs USB transactions under the programmed control of the host processor.
2 RESUME	Resume Enables USBFS to execute resume signaling when this field becomes 1. This allows USBFS to perform remote wake-up. You must write 1 to this field for the required time and then write 0 to this field. If CTL[HOSTMODEEN] becomes 1, USBFS appends a low-speed end-of-packet to the Resume signaling when this field becomes 0.
1 ODDRST	Odd Reset Resets all the BDT ODD ping/pong fields to 0 when you write 1 to this field, which specifies the EVEN BDT bank.
0 USBENSOFEN	USB Enable When Device mode is enabled, writing 1 to this field enables USBFS to operate. Writing 0 to this field disables USBFS. Writing 1 to this field causes the SIE to reset all of its STAT[ODD] to the BDTs and also resets most of the logic in the SIE. When Host mode is enabled, writing 0 to this field causes the SIE to stop sending SOF tokens. 0b - Disable 1b - Enable

61.7.2.16 Address (ADDR)

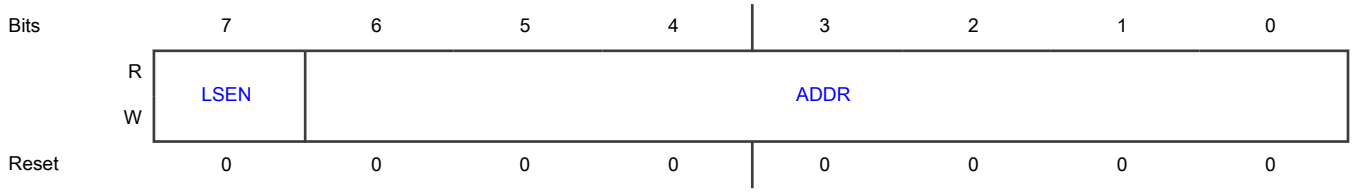
Offset

Register	Offset
ADDR	98h

Function

- In Peripheral mode (CTL[HOSTMODEEN] = 0), this register holds the unique USB address that USBFS decodes. In Host mode (CTL[HOSTMODEEN] = 1), USBFS transmits this USB address with a TOKEN packet. This enables USBFS to uniquely address any USB peripheral. In both Peripheral or Host mode: CTL[USBENSOFEN] must be 1.
- After the reset input becomes active or USBFS decodes a USB reset signal, this register resets to 00h. This action initializes this register to decode address 00h, as required by the USB specification.

Diagram



Fields

Field	Function
7 LSEN	Low Speed Enable Informs USBFS that you must perform the next token command written to Token (TOKEN) at low-speed. This enables USBFS to perform the preamble required for low-speed data transmissions.
6-0 ADDR	USB Address Defines the USB address that USBFS decodes in Peripheral mode, or that USBFS transmits when in Host mode.

61.7.2.17 BDT Page 1 (BDTPAGE1)

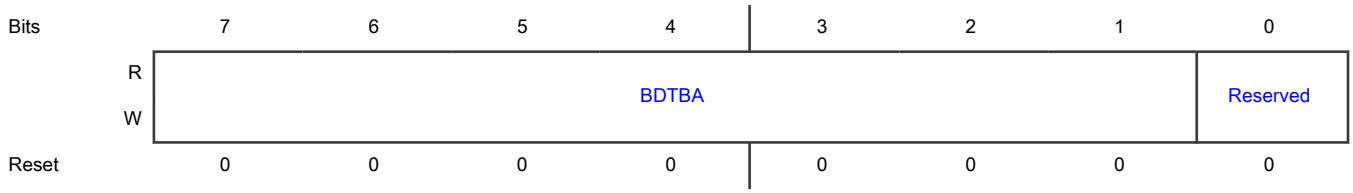
Offset

Register	Offset
BDTPAGE1	9Ch

Function

Provides address bits 15 through 9 of the base address where the current BDT resides in system memory. See Buffer descriptor table (BDT) for more information. The 32-bit BDT base address is always aligned on 512-byte boundaries, so bits 8 through 0 of the base address are always zero.

Diagram



Fields

Field	Function
7-1 BDTBA	BDT Base Address Provides address bits 15 through 9 of the BDT base address.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 —	Reserved

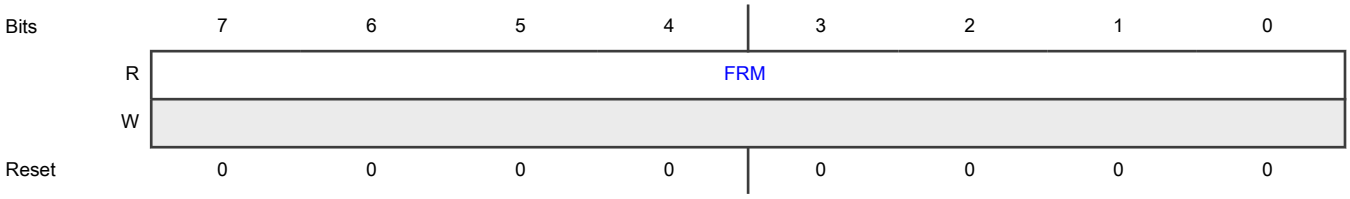
61.7.2.18 Frame Number Register Low (FRMNUML)

Offset

Register	Offset
FRMNUML	A0h

Function
Works with Frame Number Register High (FRMNUMH) to indicate the 11-bit frame number that determines the address of the BDT in system memory. When you receive ISTAT[SOFTOK], update these registers with the current frame number.

Diagram



Fields

Field	Function
7-0 FRM	Frame Number, Bits 0-7

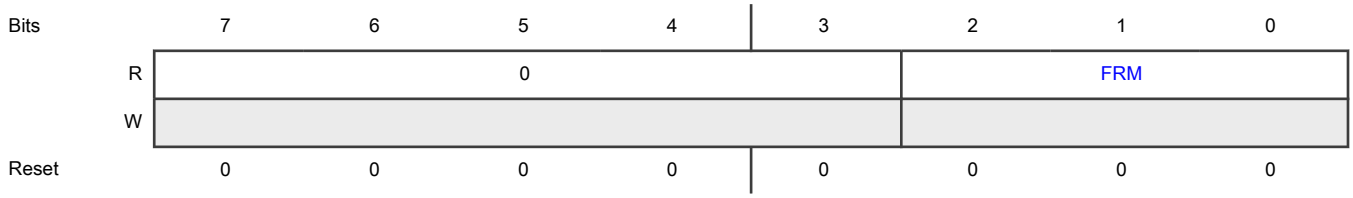
61.7.2.19 Frame Number Register High (FRMNUMH)

Offset

Register	Offset
FRMNUMH	A4h

Function
Works with Frame Number Register Low (FRMNUML) to indicate the 11-bit frame number that determines the address of the BDT in system memory. These registers are updated with the current frame number when you receive ISTAT[SOFTOK].

Diagram



Fields

Field	Function
7-3 —	Reserved
2-0 FRM	Frame Number, Bits 8-10

61.7.2.20 Token (TOKEN)

Offset

Register	Offset
TOKEN	A8h

Function

Initiates USB transactions when in Host mode (CTL[HOSTMODEEN] = 1).

When you execute a USB transaction to a peripheral, it writes the TOKEN type and endpoint to Token (TOKEN). After you wrote to Token (TOKEN), USBFS begins the specified USB transaction to the address contained in the address register.

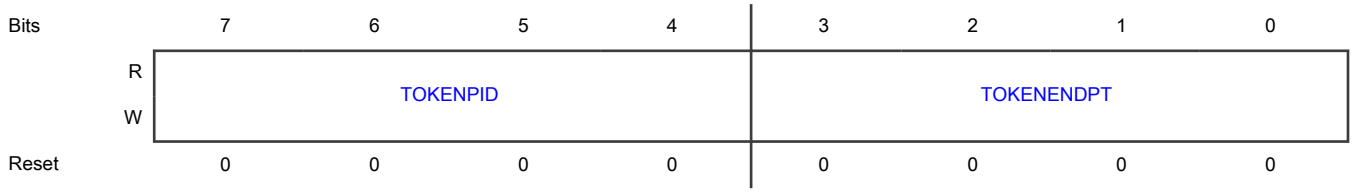
Before writing to Token (TOKEN), the processor core must always check that the TOKEN_BUSY in the control register is not 1. This ensures that the token commands are not overwritten before you can execute them.

Use Address (ADDR) and Endpoint Control (ENDPT0 - ENDPT15) when performing a token command and must write before Token (TOKEN).

Address (ADDR) selects the USB peripheral address, which the token command transmits.

Endpoint Control (ENDPT0 - ENDPT15) determines the handshake and retries policies used during the transfer.

Diagram



Fields

Field	Function
7-4 TOKENPID	Token Type Specifies the token type that USBFS executes. 0001b - OUT token. USBFS performs an OUT (TX) transaction. 1001b - IN token. USBFS performs an IN (RX) transaction. 1101b - SETUP token. USBFS performs a SETUP (TX) transaction
3-0 TOKENENDPT	Token Endpoint Address Specifies the endpoint address for the token command. The 4-bit value written must be a valid endpoint.

61.7.2.21 SOF Threshold (SOFTHLD)

Offset

Register	Offset
SOFTHLD	ACh

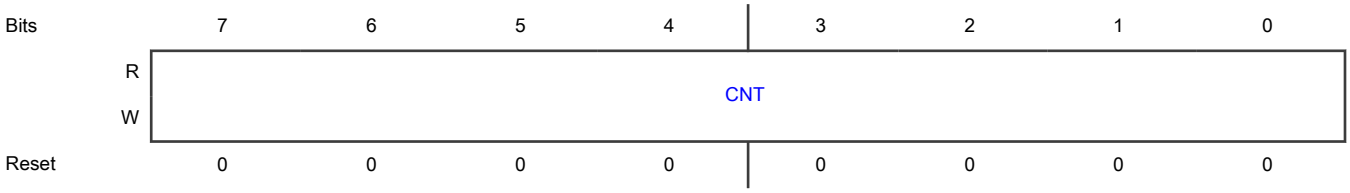
Function

This register is used only in Host mode (CTL[HOSTMODEEN] = 1). The 14-bit SOF counter counts the interval between SOF frames when in Host mode. You must transmit the SOF every 1 ms, so load the SOF counter with a value of 12000. When the SOF counter reaches zero, a SOF token is transmitted. This register programs the number of USB byte times when MISCCTRL[SOFDYNTHLD] = 0 (or 8-byte times when MISCCTRL[SOFDYNTHLD] = 1), before the SOF stops initiating token packet transactions.

Write a value to SOFTHLD[CNT] that ensures that other packets are not actively transmitted when the SOF time counts to zero. After the SOF counter reaches the threshold value, no more tokens are transmitted until after the SOF has been transmitted. The value programmed into the threshold register must reserve enough time to ensure that the worst-case transaction finishes. In general, the worst-case transaction is an IN token, followed by a data packet from the peripheral device and the host's response. The actual time required is a function of the maximum packet size on the bus. Typical values for the SOF threshold are:

- 74 for 64-byte packets
- 42 for 32-byte packets
- 26 for 16-byte packets
- 18 for 8-byte packets

Diagram



Fields

Field	Function
7-0 CNT	SOF Count Threshold Represents the SOF count threshold in byte times when MISCCTRL[SOFDYNTHLD] = 0, or 8-byte times when MISCCTRL[SOFDYNTHLD] = 1.

61.7.2.22 BDT Page 2 (BDTPAGE2)

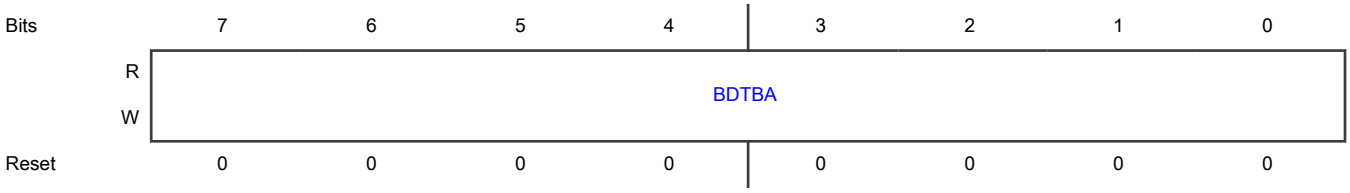
Offset

Register	Offset
BDTPAGE2	B0h

Function

Works with BDT Page 3 (BDTPAGE3) to specify address where the current BDT resides in system memory. See Buffer descriptor table (BDT) for more information.

Diagram



Fields

Field	Function
7-0 BDTBA	BDT Base Address Provides address bits 23 through 16 of the BDT base address in system memory.

61.7.2.23 BDT Page 3 (BDTPAGE3)

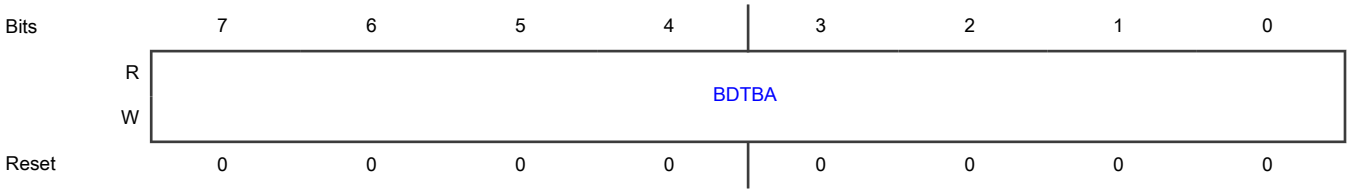
Offset

Register	Offset
BDTPAGE3	B4h

Function

Works with BDT Page 2 (BDTPAGE2) to specify the address where the current BDT resides in system memory. See Buffer descriptor table (BDT) for more information.

Diagram



Fields

Field	Function
7-0 BDTBA	BDT Base Address Provides address bits 31 through 24 of the BDT base address in system memory.

61.7.2.24 Endpoint Control (ENDPT0 - ENDPT15)

Offset

For a = 0 to 15:

Register	Offset
ENDPTa	C0h + (a × 4h)

Function

Contains the endpoint control fields for each of the 16 endpoints available within USBFS for a decoded address. The format for these registers is shown in the following figure.

- Endpoint 0 (ENDPT0) is associated with control pipe 0, which is required for all USB functions. Therefore, after a USBRST interrupt occurs, the processor core must write 1 to ENDPT0 to contain 0Dh. In Host mode, ENDPT0 determines the handshake, retry, and low-speed characteristics of the host transfer. For ENDPT0 in Host mode, common values are 4Dh for control, bulk, and interrupt transfers and 4Ch for isochronous transfers.
- For control, bulk, and interrupt transfers, ENDPTn[EPHSHK] = 1. For isochronous transfers, ENDPTn[EPHSHK] = 0.
- ENDPTn[EPCTLDIS], ENDPTn[EPRXEN], and ENDPTn[EPTXEN] define if an endpoint is enabled and the endpoint's direction. The endpoint enables and direction control is defined in the following table.

Table 485. Endpoint enable and direction control

EPCTLDIS	EPRXEN	EPTXEN	Endpoint enable and direction control
X	0	0	Disable endpoint
X	0	1	Enable endpoint for transit transfers only
X	1	0	Enable endpoint for receive transfers only
1	1	1	Enable endpoint for receive and transit transfers
0	1	1	Enable endpoint for receive and transit as well as control (SETUP) transfers.

Diagram

Bits	7	6	5	4	3	2	1	0
R	HOSTWOH...	RETRYDIS	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 HOSTWOHUB	Host Without A Hub Specifies whether a low-speed device connects to the host using a hub. This field: • Is meaningful only in Host mode. • Exists only in endpoint 0, and is reserved and read-only for all other ENDPT*n* registers. 0b - Connected using a hub (USBFS generates PRE_PID as required) 1b - Connected directly to host without a hub, or was used to attach
6 RETRYDIS	Retry Disable Specifies whether the host retries the NAK'ed transactions. This field: • Is meaningful only in Host mode. • Exists in only in ENDPT0, and is reserved and read-only for all other ENDPTn registers. When the host attempts to poll an interrupt endpoint, this field must be 1. 0b - Retried NAK'ed transactions in hardware. 1b - Do not retry NAK'ed transactions. When a transaction is NAK'ed, the BDT PID field is updated with the NAK PID, and the TOKEN_DNE interrupt becomes 1.
5 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 EPCTLDIS	Control Transfer Disable Disables control (SETUP) transfers. When this field is 0, control transfers are enabled only if ENDPTn[EPRXEN] and ENDPTn[EPTXEN] are both 1. See Table 485. 0b - Enable 1b - Disable
3 EPRXEN	Endpoint for RX transfers enable Enables the endpoint for RX transfers. See Table 485.
2 EPTXEN	Endpoint for TX transfers enable Enables the endpoint for TX transfers. See Table 485.
1 EPSTALL	Endpoint Stalled Indicates that the endpoint is stalled. • This field has priority over all other control fields in this register, but this field is only valid if ENDPTn[EPTXEN] = 1 or ENDPTn[EPRXEN] = 1. • Any access to this endpoint causes USBFS to return a STALL handshake. • After an endpoint is stalled, the host controller must intervene.
0 EPHSHK	Endpoint Handshaking Enable Enables an endpoint to perform handshaking during a transaction to this endpoint when this field becomes 1. This field is generally 1 unless the endpoint is isochronous.

61.7.2.25 USB Control (USBCTRL)

Offset

Register	Offset
USBCTRL	100h

Function

Configures:

- The USB FS transceiver
- LPUART over USB pin mode
- DP and DM lane reversal

Diagram

Bits	7	6	5	4	3	2	1	0
R	SUSP	PDE	UARTCHLS	UARTSEL	HOST_LS...	DPDM_LA...	0	
W								
Reset	1	1	0	0	0	0	0	0

Fields

Field	Function
7 SUSP	Suspend Places the USB transceiver into Suspend state. 0b - Not in Suspend state 1b - In Suspend state
6 PDE	Pulldown Enable Enables the weak pulldowns on the USB transceiver. 0b - Disable on D+ and D– 1b - Enable on D+ and D–
5 UARTCHLS	UART Signal Channel Select Selects the UART signal channel. This field is valid only when you select to use USB signals as UART signals. 0b - USB DP and DM signals are used as UART TX/RX. 1b - USB DP and DM signals are used as UART RX/TX.
4 UARTSEL	UART Select Selects USB external package pins to be used as UART signals. You must not change this field when USB data communication is active. 0b - USB DP and DM external package pins are used for USB signaling. 1b - USB DP and DM external package pins are used for UART signaling.
3 HOST_LS_EOP	Host-Mode-Only Low-Speed Device EOP Signaling Specifies whether the host drives resume End of Packet (EOP) signaling for a full-speed device or a directly-connected low-speed device. Host mode resumes the EOP signaling setting for a low-speed device. Use this field only during Host mode. 0b - Full-speed device or a low-speed device through a hub 1b - Directly-connected low-speed device
2 DPDM_LANE_REVERSE	DP and DM Lane Reversal Control Specifies an external USB DP and DM package pin assignment configuration for your flexibility in board design.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	You must not change this field when USB data communication is active. 0b - Standard USB DP and DM package pin assignment 1b - Reverse roles of USB DP and DM package pins
1-0 —	Reserved

61.7.2.26 USB OTG Observe (OBSERVE)

Offset

Register	Offset
OBSERVE	104h

Function

Indicates the state of the pullups and pulldowns at the transceiver, which is useful when interfacing with an external OTG control module via a serial interface.

Diagram

Bits	7	6	5	4	3	2	1	0
R	DPPU	DPPD	0	DMPD	0			
W								
Reset	0	1	0	1	0	0	0	0

Fields

Field	Function
7 DPPU	D+ Pullup Indicates the state of the D+ pullup enable at the USB transceiver. 0b - Disabled 1b - Enabled
6 DPPD	D+ Pulldown Indicates the state of the D+ pulldown enable at the USB transceiver. 0b - Disabled 1b - Enabled
5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
4 DMPD	D- Pulldown Indicates the state of the D- pulldown enable at the USB transceiver. 0b - Disabled 1b - Enabled
3-0 —	Reserved

61.7.2.27 USB OTG Control (CONTROL)

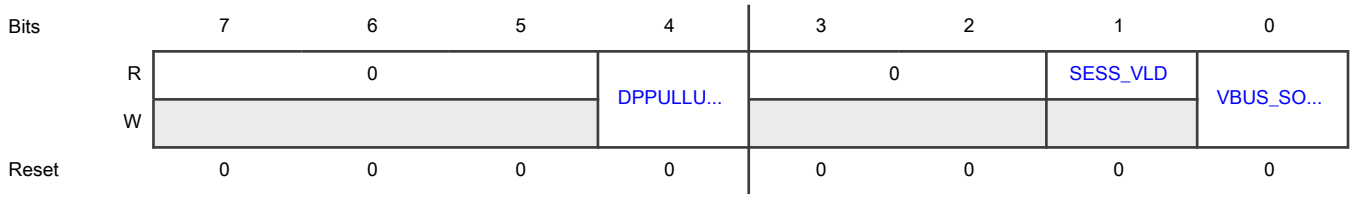
Offset

Register	Offset
CONTROL	108h

Function

Controls VBUS monitoring and the Device-mode DP pullup in the USBFS transceiver.

Diagram



Fields

Field	Function
7-5 —	Reserved
4 DPPULLUPNONOTG	DP Pullup in Non-OTG Device Mode Provides control of the DP pullup in USB OTG, if you configure USB in a non-OTG device mode. 0b - Disable 1b - Enabled
3-2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1 SESS_VLD	VBUS Session Valid status Indicates whether the VBUS monitoring method reports that VBUS is above or below the session-valid threshold. CONTROL[VBUS_SOURCE_SEL] specifies which of the following options USBFS uses to monitor the VBUS voltage: • The input to the 3.3-5 V USB regulator • A resistive divider between VBUS and the ground connected to a GPIO pin 0b - Below 1b - Above
0 VBUS_SOURCE_SEL	VBUS Monitoring Source Select Specifies the source of the VBUS monitoring signal. This source determines the value of CONTROL[SESS_VLD]. 0b - Reserved 1b - Resistive divider attached to a GPIO pin

61.7.2.28 USB Transceiver Control 0 (USBTRC0)

Offset

Register	Offset
USBTRC0	10Ch

Function

Controls the following operational aspects that the USB Full-Speed Controller registers do not provide:

- Basic operation of the on-chip USB full-speed transceiver
- USB data connection

Diagram

Bits	7	6	5	4	3	2	1	0
R		VREGIN_...	USBRESM...	VFEDG_D...	VREDG_D...	USB_CLK...	SYNC_DET	USB_RES...
W	USBRESET							
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 USBRESET	USB Reset Generates a hard reset to USB OTG. After this field becomes 1 and the reset occurs, this field becomes 0 automatically. **NOTE** **This field always reads 0. After writing 1 to this field, wait for 2 USB clock cycles before accessing other USB register fields.** 0b - Normal USBFS operation 1b - Returns USBFS to its reset state
6 VREGIN_STS	VREGIN Status Indicates the VREGIN status.
5 USBRESMEN	Asynchronous Resume Interrupt Enable Enables the USB asynchronous wake-up from Suspend mode. When this field becomes 1, after detection of resume signaling on the USB bus, it allows USBFS to send an asynchronous wake-up event to the MCU. The MCU then re-enables clocks to USBFS. This is used for low-power Suspend mode when USB module clocks are stopped or when the USB transceiver is in Suspend mode. Async wake-up only works in Device mode. • The asynchronous resume interrupt differs from the synchronous resume interrupt in that it asynchronously detects K-state using the unfiltered state of the D+ and D– pins. • You must only enable the asynchronous resume interrupt after you suspend the transceiver. 0b - Disable 1b - Enable
4 VFEDG_DET	VREGIN Falling Edge Interrupt Detect Detects the VREGIN falling edge interrupt. To enable this field, use USB *x* _MISCCTRL[VFEDG_EN]. 0b - Not detected 1b - Detected
3 VREDG_DET	VREGIN Rising Edge Interrupt Detect Detects the VREGIN rising edge interrupt.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	To enable this field, use USB *x* _MISCCTRL[VREDG_EN]. 0b - Not detected 1b - Detected
2 USB_CLK_RECOVERY_INT	Combined USB Clock Recovery interrupt status The read-only USB_CLK_RECOVERY_INT field will be set to 1'b1 when any of USB clock recovery interrupt conditions are detected and those interrupts are unmasked. • The only unmasked USB clock recovery interrupt condition results from an overflow of the frequency trim setting values, indicating that the frequency trim calculated is out of the adjustment range of the FIRC output clock. • To clear USB_CLK_RECOVERY_INT bit after it has been set, write 0xFF to register USB_CLK_RECOVER_INT_STATUS.
1 SYNC_DET	Synchronous USB Interrupt Detect Detects the synchronous USB interrupt. 0b - Not detected 1b - Detected
0 USB_RESUME_INT	USB Asynchronous Interrupt Indicates whether the interrupt is generated. 0b - Not generated 1b - Generated because of the USB asynchronous interrupt

61.7.2.29 Frame Adjust (USBFRMADJUST)

Offset

Register	Offset
USBFRMADJUST	114h

Function

Allows adjustment of frame counting interval for the Host mode operation.

Diagram

Bits	7	6	5	4	3	2	1	0
R	ADJ							
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7-0 ADJ	Frame Adjustment Specifies that in Host mode, the frame adjustment is a two's complement number that adjusts the period of each USB frame in 12 MHz clock periods. A SOF is normally generated every 12,000 cycles of a 12 MHz clock. Frame Adjust (USBFRMADJUST) can adjust this by -128 to +127, to compensate for inaccuracies in the USB 48 MHz clock. Changes to this field take effect at the start of the next frame.

61.7.2.30 Keep Alive Mode Control (KEEP_ALIVE_CTRL)

Offset

Register	Offset
KEEP_ALIVE_CTRL	124h

Function

Configures and monitors the keep-alive feature for low-power Device mode applications.

Diagram

Bits	7	6	5	4	3	2	1	0
R	WAKE_IN...	KEEP_AL...	0	WAKE_IN...	WAKE_RE...	STOP_AC...	OWN_OVE...	KEEP_AL...
W	W1C							
Reset	0	0	0	0	1	0	0	0

Fields

Field	Function
7 WAKE_INT_STS	Wakeup Interrupt Status Flag Becomes 1 if KEEP_ALIVE mode is enabled and you receive a SETUP token or software assigned token only after the token is processed and the USB state is IDLE. **NOTE** This field behaves differently for register reads and writes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When reading 0b - Interrupt did not occur 1b - Interrupt occurred When writing 0b - No effect 1b - Clear the flag
6 KEEP_ALIVE_STS	Keep Alive Status Indicates if USB is in Keep Alive mode. 0b - Not in Keep Alive mode 1b - In Keep Alive mode
5 —	Reserved
4 WAKE_INT_EN	Wakeup Interrupt Enable Enables KEEP_ALIVE_CTRL[WAKE_INT_STS] when you write 1 to this field.
3 WAKE_REQ_EN	Wakeup Request Enable Enables the USB bus wakeup request. During KEEP_ALIVE mode, bus access by the USB controller to a memory location outside the USB SRAM causes the bus access to stall until KEEP_ALIVE mode exits. When this field becomes 1, bus access by the USB controller to a memory location outside the USB SRAM generates a wakeup, allowing the bus access to complete. After the bus access finishes, the USB reenters the KEEP_ALIVE mode if no other wakeups are detected. **NOTE** An interrupt is not generated as a result of the wakeup request. 0b - Disable 1b - Enable
2 STOP_ACK_DLY_EN	Stop Acknowledge Delay Enable Has multiple functions: • When this field becomes 1, then the module responds to the ipg_stop when the USB core state is idle and there is no AHB transfer. • When this becomes 0, then the module responds to the ipg_stop when there is no AHB transfer. 0b - Enter KEEP_ALIVE mode immediately when there is no USB AHB transfer. 1b - Enter KEEP_ALIVE mode until the USB core is idle and there is no USB AHB transfer.
1	OWN Bit Override Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
OWN_OVERRD_EN	Enables OWN field override. During KEEP_ALIVE mode, when this field becomes 1, and if the received token is not SETUP, the OWN field of current BD is forced to 0, so the USB core responds with NAK.
0 KEEP_ALIVE_EN	Keep Alive Mode Enable Enables USB_KEEP_ALIVE mode. 0b - Everything remains same as before. 1b - USB shall enter USB_KEEP_ALIVE mode after asserting ipg_stop.

61.7.2.31 Keep Alive Mode Wakeup Control (KEEP_ALIVE_WKCTRL)

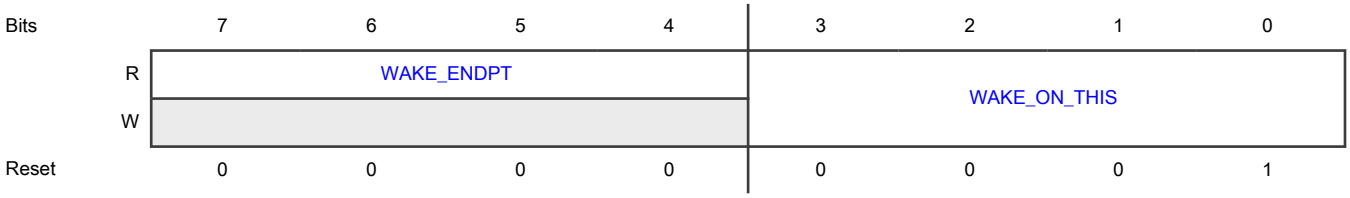
Offset

Register	Offset
KEEP_ALIVE_WKCTRL	128h

Function

Provides additional configuration and status fields for Keep Alive mode.

Diagram



Fields

Field	Function
7-4 WAKE_ENDPT	Endpoint address for the wakeup request Indicates which endpoint causes the wakeup interrupt.
3-0 WAKE_ON_THIS	Token PID for the wakeup request Specifies which token can wakeup USB during KEEP_ALIVE mode. 0001b - Wake up after receiving OUT or SETUP token packet. 1101b - Wake up after receiving SETUP token packet. All other values are reserved.

61.7.2.32 Miscellaneous Control (MISCCTRL)

Offset

Register	Offset
MISCCTRL	12Ch

Function

Provides additional controls for the USBFS subsystem.

Diagram

Bits	7	6	5	4	3	2	1	0
R	STL_ADJ...	Reserved		VFEDG_EN	VREDG_EN	OWNERRI...	SOFBUSS...	SOFDYNT...
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 STL_ADJ_EN	USB Peripheral Mode Stall Adjust Enable Specifies that this field is valid only in Peripheral mode (CTL[HOSTMODEEN] = 0). By default (STL_ADJ_EN = 0), when you stall an endpoint (ENDPTn[END_STALL] = 1), both IN and OUT directions of the endpoint are stalled. If this field is 1, then when an endpoint is stalled (ENDPTn[END_STALL] = 1), then the USBx_STALL_xx_DIS registers can control which endpoint direction(s) are affected. 0b - If ENDPTn[END_STALL] = 1, both IN and OUT directions for the associated endpoint stalls. 1b - If ENDPTn[END_STALL] = 1, the STALL_xx_DIS registers control which directions for the associated endpoint stalls.
6-5 —	Reserved
4 VFEDG_EN	VREGIN Falling Edge Interrupt Enable Enables VREGIN falling edge interrupt. 0b - Disable 1b - Enable
3 VREDG_EN	VREGIN Rising Edge Interrupt Enable Enables VREGIN rising edge interrupt. 0b - Disable 1b - Enable
2	OWN Error Detect for ISO IN and ISO OUT Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
OWNERRISODIS	Disables OWN error detect for ISO IN, and ISO out. This field is only valid for Peripheral mode (CTL[HOSTMODEEN] = 0). 0b - Enable 1b - Disable
1 SOFBUSSET	SOF_TOK Interrupt Generation Mode Select Specifies the behavior that causes the SOF_TOK interrupt to become 1. This field is meaningful only in Host mode (CTL[HOSTMODEEN] = 1). 0b - According to the SOF threshold value 1b - When the SOF counter reaches 0
0 SOFDYNTHLD	Dynamic SOF Threshold Compare mode Specifies the behavior that causes the SOF_TOK interrupt to become 1. This field is meaningful only in Host mode (CTL[HOSTMODEEN] = 1). 0b - When the byte-times SOF threshold is reached 1b - When 8 byte-times SOF threshold is reached or overstepped

61.7.2.33 Peripheral Mode Stall Disable for Endpoints 7 to 0 in IN Direction (STALL_IL_DIS)

Offset

Register	Offset
STALL_IL_DIS	130h

Function

Meaningful only in Peripheral mode (CTL[HOSTMODEEN] = 0) and when MISCCTRL[STL_ADJ_EN] = 1.

When you stall an endpoint (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the IN direction of the endpoints 7 to 0.

Diagram

Bits	7	6	5	4	3	2	1	0
R W	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 STALL_I_DIS7	Disable Endpoint 7 IN Direction Disables USB Device mode stall for endpoint 7 IN direction. 0b - Enable 1b - Disable
6 STALL_I_DIS6	Disable Endpoint 6 IN Direction Disables USB Device mode stall for endpoint 6 IN direction. 0b - Enable 1b - Disable
5 STALL_I_DIS5	Disable Endpoint 5 IN Direction Disables USB Device mode stall for endpoint 5 IN direction. 0b - Enable 1b - Disable
4 STALL_I_DIS4	Disable Endpoint 4 IN Direction Disables USB Device mode stall for endpoint 4 IN direction. 0b - Enable 1b - Disable
3 STALL_I_DIS3	Disable Endpoint 3 IN Direction Disables USB Device mode stall for endpoint 3 IN direction. 0b - Enable 1b - Disable
2 STALL_I_DIS2	Disable Endpoint 2 IN Direction Disables USB Device mode stall for endpoint 2 IN direction. 0b - Enable 1b - Disable
1 STALL_I_DIS1	Disable Endpoint 1 IN Direction Disables USB Device mode stall for endpoint 1 IN direction.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enable 1b - Disable
0 STALL_I_DIS0	Disable Endpoint 0 IN Direction Disables USB Device mode stall for endpoint 0 IN direction. 0b - Enable 1b - Disable

61.7.2.34 Peripheral Mode Stall Disable for Endpoints 15 to 8 in IN Direction (STALL_IH_DIS)

Offset

Register	Offset
STALL_IH_DIS	134h

Function

Meaningful only in Peripheral mode (CTL[HOSTMODEEN] = 0) and when MISCCTRL[STL_ADJ_EN] = 1.

When you stall an endpoint (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the IN direction of the endpoints 15 to 8.

Diagram

Bits	7	6	5	4	3	2	1	0
R W	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...	STALL_I...
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 STALL_I_DIS15	Disable Endpoint 15 IN Direction Disables USB Device mode stall for endpoint 15 IN direction. 0b - Enable 1b - Disable
6 STALL_I_DIS14	Disable Endpoint 14 IN Direction Disables USB Device mode stall for endpoint 14 IN direction.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enable 1b - Disable
5 STALL_I_DIS13	Disable Endpoint 13 IN Direction Disables USB Device mode stall for endpoint 13 IN direction. 0b - Enable 1b - Disable
4 STALL_I_DIS12	Disable Endpoint 12 IN Direction Disables USB Device mode stall for endpoint 12 IN direction. 0b - Enable 1b - Disable
3 STALL_I_DIS11	Disable Endpoint 11 IN Direction Disables USB Device mode stall for endpoint 11 IN direction. 0b - Enable 1b - Disable
2 STALL_I_DIS10	Disable Endpoint 10 IN Direction Disables USB Device mode stall for endpoint 10 IN direction. 0b - Enable 1b - Disable
1 STALL_I_DIS9	Disable Endpoint 9 IN Direction Disables USB Device mode stall for endpoint 9 IN direction. 0b - Enable 1b - Disable
0 STALL_I_DIS8	Disable Endpoint 8 IN Direction Disables USB Device mode stall for endpoint 8 IN direction. 0b - Enable 1b - Disable

61.7.2.35 Peripheral Mode Stall Disable for Endpoints 7 to 0 in OUT Direction (STALL_OL_DIS)

Offset

Register	Offset
STALL_OL_DIS	138h

Function

Meaningful only in Peripheral mode (CTL[HOSTMODEEN] = 0) and when MISCCTRL[STL_ADJ_EN] = 1.

When you stall an endpoint (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the OUT direction for the endpoints 7 to 0.

Diagram

Bits	7	6	5	4	3	2	1	0
R W	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 STALL_O_DIS7	Disable Endpoint 7 OUT Direction Disables USB Device mode stall for endpoint 7 OUT direction. 0b - Enable 1b - Disable
6 STALL_O_DIS6	Disable Endpoint 6 OUT Direction Disables USB Device mode stall for endpoint 6 OUT direction. 0b - Enable 1b - Disable
5 STALL_O_DIS5	Disable Endpoint 5 OUT Direction Disables USB Device mode stall for endpoint 5 OUT direction. 0b - Enable 1b - Disable
4 STALL_O_DIS4	Disable Endpoint 4 OUT Direction Disables USB Device mode stall for endpoint 4 OUT direction. 0b - Enable 1b - Disable
3 STALL_O_DIS3	Disable Endpoint 3 OUT Direction Disables USB Device mode stall for endpoint 3 OUT direction. 0b - Enable 1b - Disable
2 STALL_O_DIS2	Disable Endpoint 2 OUT Direction Disables USB Device mode stall for endpoint 2 OUT direction.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enable 1b - Disable
1 STALL_O_DIS1	Disable Endpoint 1 OUT Direction Disables USB Device mode stall for endpoint 1 OUT direction. 0b - Enable 1b - Disable
0 STALL_O_DIS0	Disable Endpoint 0 OUT Direction Disables USB Device mode stall for endpoint 0 OUT direction. 0b - Enable 1b - Disable

61.7.2.36 Peripheral Mode Stall Disable for Endpoints 15 to 8 in OUT Direction (STALL_OH_DIS)

Offset

Register	Offset
STALL_OH_DIS	13Ch

Function

Meaningful only in Peripheral mode (CTL[HOSTMODEEN] = 0) and when MISCCTRL[STL_ADJ_EN] = 1.

When you stall an endpoint (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the OUT direction for the endpoints 15 to 8.

Diagram

Bits	7	6	5	4	3	2	1	0
R W	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...	STALL_O...
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 STALL_O_DIS15	Disable Endpoint 15 OUT Direction Disables USB Device mode stall for endpoint 15 OUT direction.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enable 1b - Disable
6 STALL_O_DIS14	Disable Endpoint 14 OUT Direction Disables USB Device mode stall for endpoint 14 OUT direction. 0b - Enable 1b - Disable
5 STALL_O_DIS13	Disable Endpoint 13 OUT Direction Disables USB Device mode stall for endpoint 13 OUT direction. 0b - Enable 1b - Disable
4 STALL_O_DIS12	Disable endpoint 12 OUT direction Disables USB Device mode stall for endpoint 12 OUT direction. 0b - Enable 1b - Disable
3 STALL_O_DIS11	Disable Endpoint 11 OUT Direction Disables USB Device mode stall for endpoint 11 OUT direction. 0b - Enable 1b - Disable
2 STALL_O_DIS10	Disable Endpoint 10 OUT Direction Disables USB Device mode stall for endpoint 10 OUT direction. 0b - Enable 1b - Disable
1 STALL_O_DIS9	Disable Endpoint 9 OUT Direction Disables USB Device mode stall for endpoint 9 OUT direction. 0b - Enable 1b - Disable
0 STALL_O_DIS8	Disable Endpoint 8 OUT Direction Disables USB Device mode stall for endpoint 0 OUT direction. 0b - Enable 1b - Disable

61.7.2.37 USB Clock Recovery Control (CLK_RECOVER_CTRL)

Offset

Register	Offset
CLK_RECOVER_CTRL	140h

Function

Controls the crystal-less USB clock mode in which the internal FIRC oscillator is tuned to match the clock extracted from the incoming USB data stream.

You must configure the FIRC internal oscillator module in the SCG module and enable SOF frequency tuning in this register and FIRC Oscillator Enable (CLK_RECOVER_IRC_EN) for this mode.

Diagram

Bits	7	6	5	4	3	2	1	0
R	CLOCK_R...	RESET_R...	RESTART...	0	TRIM_IN...	Reserved	Reserved	Reserved
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7 CLOCK_RECOVER_EN	Crystal-Less USB Enable Enables the frequency trimming logic for the FIRC module to generate clocks for crystal-less FS USB Device mode operation. You must initialize the FIRCCSR[FIRCEN] and FIRCTCFG[TRIMSRC] to enable the FIRC module itself. Then this field and CLK_RECOVER_IRC_EN[IRC_EN] must become 1 before using the crystal-less USB clock configuration. **NOTE** **You must not write 1 to this field for USB Host mode or OTG operation.** 0b - Disable 1b - Enable
6 RESET_RESUME_ROUGH_EN	Reset or Resume to Rough Phase Enable Indicates that the clock recovery block tracks the FIRC to get an accurate 48 MHz clock. It has 2 phases after you enable CLK_RECOVER_CTRL[CLOCK_RECOVER_EN], rough phase and tracking phase. The step to fine-tune the FIRC by adjusting the trim fine value differs during these 2 phases. The rough step-in phase is larger than that in the tracking phase. Switch back to the rough stage whenever a USB bus reset or bus resume occurs. 0b - Always works in tracking phase after the first time rough phase, to track transition. 1b - Go back to rough stage whenever a bus reset or bus resume occurs.
5	Restart from IFR Trim Value

Table continues on the next page...

Table continued from the previous page...

Field	Function
RESTART_IFR TRIM_EN	FIRC has a default trim fine value whose default value is factory trimmed (the IFR trim value). The clock recover block tracks the clock's accuracy at 48 MHz and keeps updating the trim fine value accordingly. 0b - Trim fine adjustment always works based on the previous updated trim fine value. 1b - Trim fine restarts from the IFR trim value whenever you detect bus_reset or bus_resume or deassert module enable.
4 —	Reserved
3 TRIM_INIT_VAL _SEL	Selects the source for the initial FIRC trim fine value used after a reset. 0b - Mid-scale 1b - IFR
2 —	Reserved Do not modify the value of this field.
1 —	Reserved
0 —	Reserved

61.7.2.38 FIRC Oscillator Enable (CLK_RECOVER_IRC_EN)

Offset

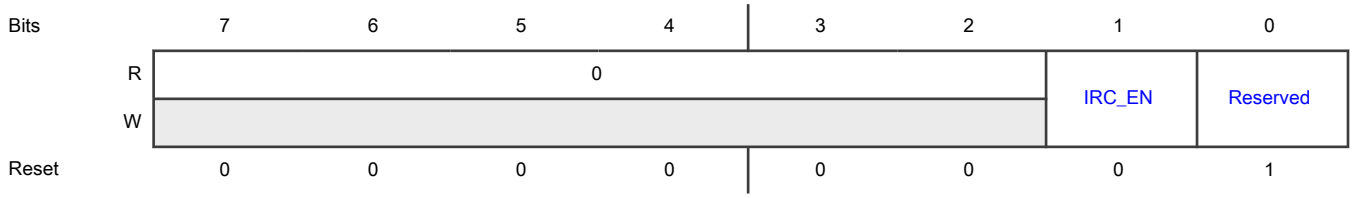
Register	Offset
CLK_RECOVER_IRC_E N	144h

Function

Controls the USB-specific frequency trimming behavior for the on-chip FIRC module used to produce nominal 48 MHz clocks for USB crystal-less operation and other functions.

See the FIRC operation in the "Clock Distribution" section in the SCG chapter for more information.

Diagram



Fields

Field	Function
7-2 —	Reserved
1 IRC_EN	Fast IRC enable Configures the frequency trimming logic for the FIRC module to generate clocks for crystal-less FS USB Device mode operation. You must initialize FIRCCSR[FIRCEN] and FIRCTCFG[TRIMSRC] for the SCG module to enable the FIRC module itself. Then this field and CLK_RECOVER_CTRL[CLOCK_RECOVER_EN] must become 1 before using the crystal-less USB clock configuration. 0b - Disable 1b - Enable
0 —	Reserved

61.7.2.39 Clock Recovery Combined Interrupt Enable (CLK_RECOVER_INT_EN)

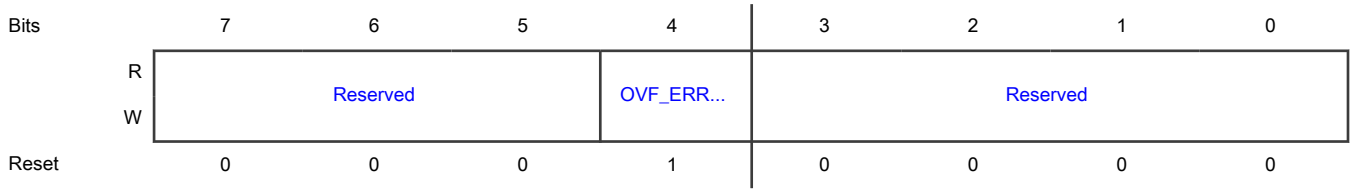
Offset

Register	Offset
CLK_RECOVER_INT_EN	154h

Function

Enables or masks the individual interrupt flags, which are logically OR'ed together, to produce the combined interrupt indication at USBTRC0[USB_CLK_RECOVERY_INT] if you detect the indicated conditions in the USB clock recovery algorithm operation.

Diagram



Fields

Field	Function
7-5 —	Reserved
4 OVF_ERROR_EN	Overflow error interrupt enable Determines whether the OVF_ERROR condition signal is used in generation of USB_CLK_RECOVERY_INT. 0b - The interrupt is masked 1b - The interrupt is enabled
3-0 —	Reserved

61.7.2.40 Clock Recovery Separated Interrupt Status (CLK_RECOVER_INT_STATUS)

Offset

Register	Offset
CLK_RECOVER_INT_STATUS	15Ch

Function

Contains flags for clock-recovery interrupts.

Diagram

Bits	7	6	5	4	3	2	1	0
R	Reserved			OVF_ERR...	Reserved			
W	W1C			W1C	W1C			
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7-5 —	Reserved
4 OVF_ERROR	Overflow Error Interrupt Status Flag Indicates that the USB clock recovery algorithm detects that the frequency trim adjustment needed for the FIRC output clock is outside the available TRIM_FINE adjustment range for FIRC. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Interrupt did not occur 1b - Unmasked interrupt occurred When writing 0b - No effect 1b - Clear the flag
3-0 —	Reserved

Chapter 62
Universal Serial Bus 2.0 High-Speed Integrated PHY (USBHS_PHY)

62.1 Chip-specific USBHS_PHY information

Table 486. Reference links to related information

Topic	Related module	Reference
Full description	USBHS_PHY	USBHS_PHY
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

See AHB Clock Control 2 (AHBCLKCTRL2) to enable USBHS_PHY clock. Also see Peripheral Reset Control 2 (PRESETCTRL2) for reset control.

62.1.1 Module instances

This device has one instance of USBHS_PHY module.

62.1.2 USBHS_PHY operation modes

USBHS_PHY is functional only in SD and OD modes. It is non-functional in MD mode. Configure SPC.ACTIVE_CFG[DCDC_VDD_LVL] = SPC.ACTIVE_CFG[CORELDO_VDD_LVL] >= 0x2 for correct operation of the module.

62.2 Overview

This chip contains a USBPHY macrocell capable of operating in either:

- Device mode: In this mode, the USB physical layer can connect to USB host systems at the USB high-speed (HS) rate of 480 Mbit/s or full-speed (FS) rate of 12 Mbit/s.
- Host mode: In this mode, USBPHY can connect to peripheral devices operating at HS, FS, or the USB 2.0 low-speed (LS) rate of 1.5 Mbit/s.

USBPHY communicates with the USB HS controller using a standard UTMI+ interface. The module includes the following for configuration and status reporting:

- 480 MHz PLL
- UTMI digital logic and state machines
- Analog transceiver circuits
- IPS bus interface

The USB_DP and USB_DM pins connect directly to a USB connector.

See the "USB Device Charger Detection module (USBDCD)" chapter for the digital and analog portions that USBPHY contains and the module functions it performs. USBPHY also supports features for detection and signaling described in *USB Battery Charging Specification, Revision 1.2*. Also see descriptions for Charger Detect (USB1_CHRG_DETECT) and Charger Detect Status (USB1_CHRG_DET_STAT) for a better insight into USBPHY's charger detection functions.

62.2.1 Block diagram

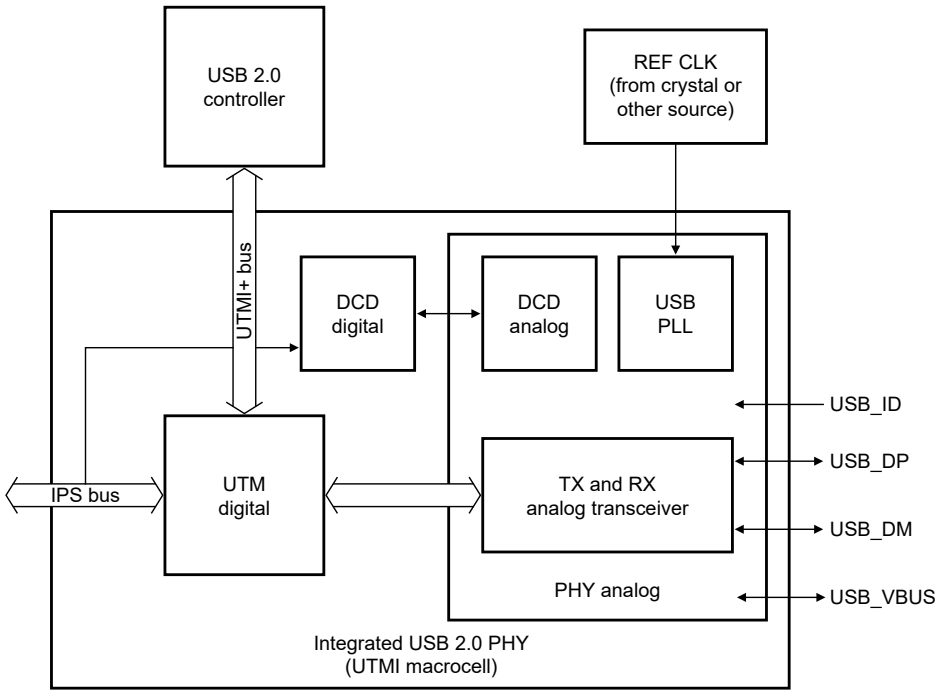


Figure 338. Block diagram

62.2.2 Features

- Supports HS and FS in Device mode
- Supports HS, FS, and LS in Host mode
- Complies with the UTMI+ standard with a 16-bit data interface to the USB controller
- Provides an integrated 480 MHz PLL
- Supports USBDCD with features for detection and signaling described in *USB Battery Charging Specification, Revision 1.2*
- Supports a programmable phase fractional divider (PFD) output clock

62.3 Functional description

62.3.1 Operation

UTMI provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz to match the controller instance on the chip:

- The digital portion of USBPHY includes:
  - UTMI interface and control and status logic
  - Digital transmitter
  - Digital receiver
  - Programmable registers
- The analog transceiver section includes the following for all three USB 2.0 signaling speeds, as shown in Figure 340:

— Analog receiver

— Analog transmitter

62.3.1.1 UTMI interface and control and status logic

The UTMI interface and control and status logic handle the UTMI+ interface signals, line_state bits, reset buffering, some suspend and resume functions, transceiver speed selection, and transceiver termination selection.

The PLL in USBPHY supplies a 480 MHz clock to the USBPHY's digital section, and the UTMI interface and control and status logic divide the clock to generate the 30 MHz clock used in the UTMI+ interface.

This 480 MHz clock is also the input to the PFD hardware module. The PFD can generate the PFD output clock frequency by using the following formula: 480 MHz × 18 ÷ *n*, where *n* = 18–35. The mux is shown in the following figure (see PFDA[PFD0_FRAC] and PFDA[PFD_CLK_SEL]).

NOTE

The maximum programmable frequency allowed for the PFD output clock, USB1PFDCLK, is chip-specific.

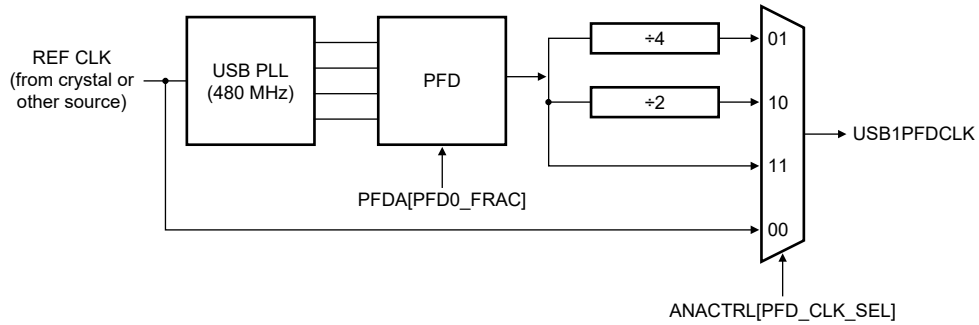


Figure 339. USB1PFDCLK generation diagram

62.3.1.2 Digital transmitter

The digital transmitter receives a 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh, and tx_ready handshakes.

In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bit stream at 480 Mbit/s for HS, 12 Mbit/s for FS, or 1.5 Mbit/s for LS. The digital transmitter does this conversion while implementing the bit-stuffing algorithm and the NRZI encoder that you use to remove the DC component from the serial bit stream. The output of this encoder is sent to the LS, FS, or HS differential drivers (see LS or FS differential driver and HS differential driver) in the transmitter block of the analog transceiver section.

62.3.1.3 Digital receiver

The digital receiver, when operating at HS, receives the 480 MHz, 9X oversampled data from the combination of HS differential receiver and 9X oversample module. When operating at LS or FS, it receives the raw serial bit stream from LS or FS differential receiver. Within the digital receiver, the LS or FS serial bit stream is oversampled 5X for the appropriate signaling rate. The oversampled data streams go to separate HS DLL and LS DLL or FS DLL circuits used for data recovery.

As the phase of the USB remote transmitter shifts relative to the local PLL, the HS DLL, LS DLL, or FS DLL tracks these changes to give a reliable sample of the incoming bit stream. Because this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. When the received data bit stream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bit stream and pass it to a deserializer and holding register.

The RX state machine handles the rx_valid and rx_validh handshakes with the USB controller. If the handshakes are not interlocked, then there is no rx_ready signal from the controller. The controller must take each 16-bit value that the USBPHY

presents. The RX state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

62.3.1.4 Analog receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module , as shown in Figure 340 and described further in this section.

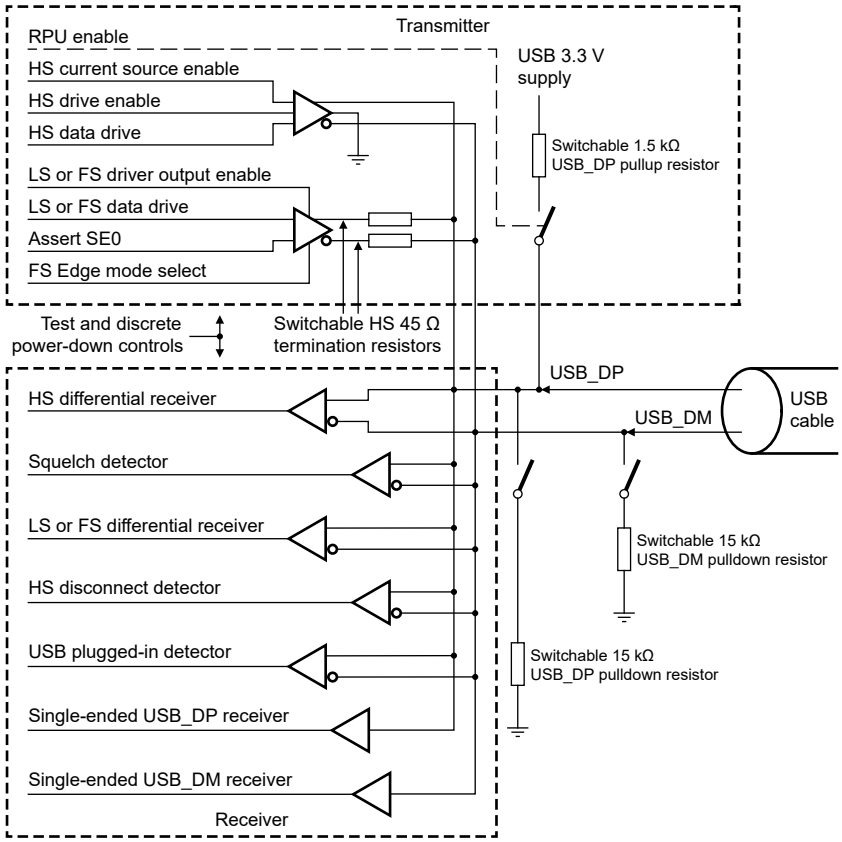


Figure 340. Analog transceiver block diagram

62.3.1.4.1 HS differential receiver

The HS differential receiver is optimized to work with the low-amplitude, ground-referenced signaling used during the HS USB operation. If the received signal voltage on the USB_DP pin is greater than the signal voltage on the USB_DM pin, the output of the receiver is high (1b). Otherwise, its output is low (0b). The purpose of the HS differential receiver is to discriminate the ± 400 mV differential voltage resulting from the current flow of high-speed drivers into the dual 45 Ω terminations found on each pin of the differential pair.

The squelch detector (see Squelch detector) ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of the common-mode offset.

62.3.1.4.2 Squelch detector

The squelch detector (also called envelope detector) is a differential analog receiver and threshold comparator. If the differential magnitude is less than a nominal 100 mV threshold, its output is high (1b). Otherwise, its output is low (0b). The purpose of the squelch detector is to invalidate HS differential receiver when the incoming signal is too low to be received reliably.

62.3.1.4.3 LS or FS differential receiver

The LS or FS differential receiver is optimized for the LS or FS signaling crossover voltage range of 1.3 V to 2.0 V. It must have an input sensitivity of better than 200 mV over a common-mode range of 0.8 V to 2.5 V.

If the signal voltage on the USB_DP pin is higher than the signal voltage on the USB_DM pin, the output of the receiver is high (1b). Otherwise, its output is low (0b). The digital receiver (see Digital receiver) decodes this output data of the receiver into J or K states, depending on the speed.

62.3.1.4.4 HS disconnect detector

The HS disconnect detector (also called the host disconnect detector) is a differential analog receiver and threshold comparator. If the differential magnitude between the USB_DP and USB_DM pins is greater than a nominal 575 mV threshold, its output is high (1b). Otherwise, its output is low (0b). The HS disconnect detector is only used in Host mode. The output of the detector is sampled during the long end-of-packet (EOP) that occurs at the end of the HS start-of-frame (SOF) packet, as enabled by CTRL[ENHOSTDISCONDETECT].

62.3.1.4.5 USB plugged-in detector

When operating in Device mode, USBPHY provides the following methods for the local USB device to detect the attachment of a cable that is also attached to a remote USB host or hub. Enable only one of these methods at a time when waiting to identify the cable attachment event, and disable all of them during USB data communication:

- Data contact detect: This is the preferred standards-based method to use the function as per *USB Battery Charging Specification, Revision 1.2*. The use of this method is described in the "USB Device Charger Detection Module (USBDCD)" chapter.
- USB plugged-in (cable attach) detector: This method looks for both USB_DP and USB_DM to be high. There is a pair of large on-chip switchable pullup resistors (200 kΩ). This pair holds both USB_DP and USB_DM high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case. When operating in Device mode, the upstream port in the remote host or hub interface contains a 15 kΩ pulldown resistor, which could easily override the 200 kΩ pullup resistor. When the cable attachment event occurs, at least one signal in the pair is low, which forces the plugged-in detector output to be high. The USB plugged-in detector is controlled through CTRL[ENDEVPLUGINDETECT] and the results are observable through STATUS[DEVPLUGIN_STATUS].

62.3.1.4.6 Single-ended USB_DP receiver

The single-ended USB_DP receiver output is high (1b) whenever the USB_DP input is above its nominal 1.4 V threshold. Otherwise, its output is low (0b).

62.3.1.4.7 Single-ended USB_DM receiver

The single-ended USB_DM receiver output is high (1b) whenever the USB_DM input is above its nominal 1.4 V threshold. Otherwise, its output is low (0b).

62.3.1.4.8 9X oversample module

The 9X oversample module uses nine identically spaced phases of the 480 MHz clock to sample the HS received bit data. The squelch signal is sampled only at 1X.

62.3.1.5 Analog transmitter

The analog transmitter comprises two differential drivers: one for HS signaling and another for LS or FS signaling. It also contains the switchable 45 Ω termination resistors (see Switchable 45 Ω termination resistors), 1.5 kΩ Device mode pullup resistor, and the switchable 15 kΩ Host mode pulldown resistors (see Figure 341).

62.3.1.5.1 Switchable 45 Ω termination resistors

The HS current-mode differential signaling requires a 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable.

Because each signal is terminated in parallel with 45 Ω at each end, each HS driver sees a 22.5 Ω DC load. This load impedance is too low for LS or FS signaling levels, generating the need for switchable HS terminating resistors. Selectable trimming resistors are provided to tune the actual termination resistance of each output pin, as shown in Figure 341. For example, TX[TXCAL45DP] allows one of the 16 trimming resistor values to be placed in parallel with the 45 Ω termination on the USB_DP signal.

These termination resistors are integrated with LS or FS differential driver and the same resistors also serve as the 45 Ω source series terminations required for LS or FS signaling.

62.3.1.5.2 LS or FS differential driver

The LS or FS differential drivers are essentially a pair of low-impedance pullup and pulldown devices that are switched in Differential mode for most LS or FS signaling. One output is driven high when the other output is driven low to signal the J or K state. Both outputs are driven low for the LS or FS SE0 states. Both the USB_DP and USB_DM LS or FS pulldown drivers are also turned on simultaneously for HS signaling. This effect causes switching in both 45 Ω terminating resistors.

62.3.1.5.3 HS differential driver

The HS differential driver receives a 17.78 mA current from the constant current source (Iref) and essentially steers it down either the USB_DP signal or USB_DM signal, or alternatively to the ground.

This current produces approximately a 400 mV drop across the 22.5 Ω termination that the driver sees when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage bandgap (Vbg) circuit.

62.3.1.5.4 Switchable 1.5 kΩ USB_DP pullup resistor

USBPHY provides a switchable 1.5 kΩ pullup resistor on the USB_DP signal. This resistor is switched on to indicate to the host or hub controller that an FS-capable device is on the USB cable, powered on, and ready. This resistor is switched off at POR, so the host does not recognize a USB device until the processor software enables the announcement of an FS device.

62.3.1.5.5 Switchable 15 kΩ USB_DP and USB_DM pulldown resistors

USBPHY provides switchable 15 kΩ pulldown resistors on both USB_DP and USB_DM signals. They are enabled in Host mode to indicate to the device controller on the far end of the USB connection that a host is present. These resistors are also used during certain battery charging detector functions.

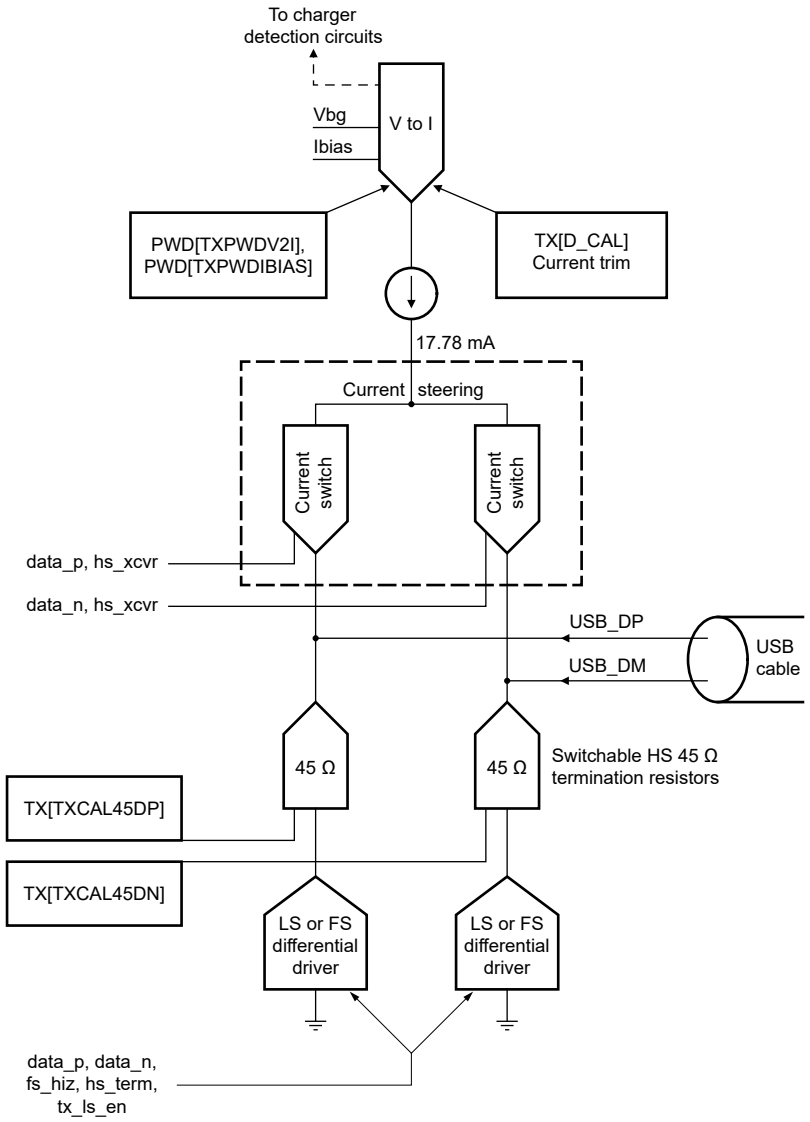


Figure 341. USBPHY transmitter block diagram

62.3.2 Low-Power mode

USBPHY includes Low-Power mode (Suspend mode) to save power consumption. When the USB bus is idle, you can place the transceiver in this mode, after which you can stop the clocks on USBPHY. The 32 kHz low-power clock must remain active as it is needed for wake-up detection.

The USB HS controller controls entry and exit into Low-Power mode:

- If USB.PORTSC1[SUSPEND] is 1, the UTMI interface is driven to the FS or LS state.
- If USB.PORTSC1[PHCD] is 1, the UTMI signal, SuspendM, goes low.
- If CTRL[ENAUTOCLR_PHY_PWD] and CTRL[ENAUTOCLR_CLKGATE] are 1, the utmi_clk and PHY transmitter are automatically enabled when the SuspendM signal goes high.
- If PLL_SIC[MISC2_CONTROL0] is 1, the PLL is powered down automatically when in Suspend mode. The PLL is automatically enabled when the SuspendM signal goes high. After the PLL is locked, USBPHY uses the PLL clock again.

Low-Power mode is always entered under the control of the driver software. See the "Universal Serial Bus Controller (USB)" chapter for details.

62.3.3 Clocking

Table 487. Input clocks

Clock	Description
ipg_clk	Free-running IPS bus clock for configuring registers
clk_xtal	24 MHz crystal clock for the PLL source
ckil_32k	32 kHz clock, which is used during Suspend mode if you need USB as a wake-up source

Table 488. Output clocks

Clock	Description
ref_pfd	PLL PFD clock that you can use as a system clock source
utmi_clk	30 MHz clock to the USB controller

62.3.4 Reset

Table 489. Resets

Reset	Description
Chip	Resets the logic and registers in USBPHY to their default states
UTMI	Resets the UTMI state machines, FIFO, and so on
Software	Soft-resets (SFTRST) USBPHY's Power Down (PWD), TX Control (TX), RX Control (RX), General Purpose Control (CTRL), STATUS[OTGID_STATUS], Debug 0 (DEBUG0), and IP Block (IP), when you write 1 to CTRL[SFTRST], along with resetting state machines in USBPHY Write 0 to CTRL[SFTRST] to release USBPHY from reset. **NOTE** A soft-reset can take multiple clock periods to complete; therefore, do not write 1 to CTRL[CLKGATE] when writing 1 to CTRL[SFTRST]. The reset process gates the clocks automatically.

62.3.5 Interrupts

Table 490. Interrupts

Flag or event	Interrupt enable	Description
CTRL[DEVPLUGIN_IRQ]	CTRL[ENIRQDEVPLUGIN]	Indicates nonstandard resistive plugged-in detection
CTRL[RESUME_IRQ]	CTRL[ENIRQRESUMEDETECT]	Indicates that the host is sending resume signaling after a Suspend signal
CTRL[WAKEUP_IRQ]	CTRL[ENIRQWAKEUP]	Indicates that the USB controller has stopped driving the SuspendM signal in its active-low state to USBPHY
CTRL[OTG_ID_CHG_IRQ]	CTRL[ENOTG_ID_CHG_IRQ]	Indicates that the value of the OTG ID pin has changed

Table continues on the next page...

Table 490. Interrupts (continued)

Flag or event	Interrupt enable	Description
CTRL[HOSTDISCONDETECT_IRQ]	CTRL[ENIRQHOSTDISCON]	Indicates that the host has detected USB device disconnection in HS mode
USBDCD interrupt	USBDCD interrupt enable	See the "Interrupts" section in the "USB Device Charger Detection Module (USBDCD)" chapter for details.

62.4 External signals

Table 491. USBPHY signals

Signal	Description	Direction
USB_DP	D+ pin	I/O
USB_DM	D– pin	I/O
USB_ID	ID Pin	Input
USB_VBUS	VBUS pin	Power

62.5 Initialization

To initialize this module, USBPHY is designed for minimal configuration while retaining significant programmability.

Table 492. USBPHY's configuration procedure

Step	Purpose	Programming
1	Release USBPHY from reset.	Write 0 to CTRL[SFTRST].
2	Run clocks.	Write 0 to CTRL[CLKGATE].
3	Reset the fields in Power Down (PWD).	Write 0 to the fields in Power Down (PWD).
4	Enable the UTMI+ level.	Write 1 to CTRL[ENUTMILEVEL2] and CTRL[ENUTMILEVEL3].
5	Enable other fields in General Purpose Control (CTRL) as needed.	Write 1 to CTRL[ENAUTOCLR_PHY_PWD], CTRL[ENAUTOCLR_CLKGATE], CTRL[AUTORESUME_EN], and so on.
6	Enable the reference clock system oscillator for the USB PLL.	Configure the reference clock system oscillator.
7	Control the USB PLL feedback loop divider.	Configure PLL_SIC[PLL_DIV_SEL] according to the system oscillator frequency.
8	Enable the USBPHY PLL regulator.	Write 1 to PLL_SIC[PLL_REG_ENABLE].
9	Clear the bypass feature.	Write 0 to PLL_SIC[PLL_BYPASS].
10	Configure other fields in PLL SIC (PLL_SIC) as needed.	Write to PLL SIC (PLL_SIC).
11	Power up the PLL.	Write 1 to PLL_SIC[PLL_POWER].
12	Enable the USB 3.3 V and 1.8 V monitors.	Write 1 to ANACTRL[LVI_EN].

Table continues on the next page...

Table 492. USBPHY's configuration procedure (continued)

Step	Purpose	Programming
13	Wait for the PLL clock lock.	Poll PLL_SIC[PLL_LOCK].
14	Enable the USB clock.	Write 1 to PLL_SIC[PLL_EN_USB_CLKS].

62.6 Application information

62.6.1 VBUS detector

USBPHY has three separate VBUS detectors, which you can individually power on or off using USB1_VBUS_DETECT[VBUSVALID_PWRUP_CMPS]. For power efficiency, align the USB software to the desired use case and either enable only one detector (if an internal detector is desired) or none of them (if an external detector is desired).

All three detectors have observable status fields in VBUS Detect Status (USB1_VBUS_DET_STAT), and to accomplish software overrides. Use the usb_vbusvalid_ext signal as a hardware override.

USB1_VBUS_DET_STAT[VBUS_VALID] is often used in Host mode; it is the most accurate and has a programmable threshold that USB1_VBUS_DETECT[VBUSVALID_THRESH] defines.

A session valid detector is often used in Device mode and may be useful in systems using nonstandard VBUS voltages.

You can use USB1_VBUS_DET_STAT[VBUS_VALID_3V] in Host mode, which has a lower threshold for the voltage on the USB1_VBUS pin than either the session valid detector or USB1_VBUS_DET_STAT[VBUS_VALID]. It may be useful for applications that provide a nonstandard VBUS voltage to USBPHY.

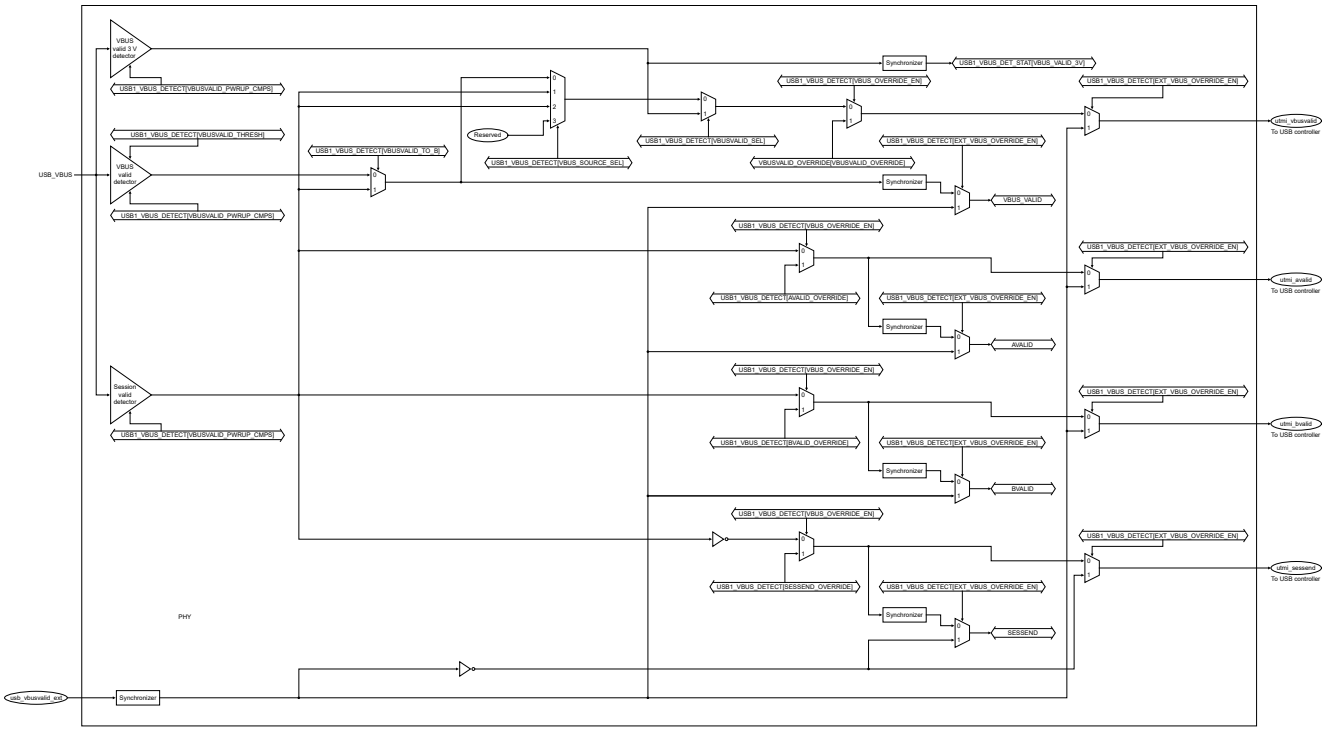


Figure 342. VBUS detectors and control registers

62.6.2 Recommended register configuration for USB certification

USBPHY has the programmability to adjust several transceiver parameters. You can:

- Adjust the TX HS driver termination impedance, FS driver output impedance, and HS driver output currents by using TX Control (TX). You can disable the ability to override the default parametric trim fields using TX Control (TX) by writing 0 to TRIM_OVERRIDE_EN[TX_CAL45DM_OVERRIDE], TRIM_OVERRIDE_EN[TX_CAL45DP_OVERRIDE], and TRIM_OVERRIDE_EN[TX_D_CAL_OVERRIDE].
- Adjust the HS RX thresholds for Squelch detector and HS disconnect detector by using RX Control (RX).

The default values of the transceiver parametric trim fields are centered, with no changes needed for USB certification testing when the chip is used with boards optimized for signal integrity on the HS USB port.

In other cases, such as when external components are inserted between the USB_DP or USB_DM pins and the USB connector, or other compromises are made on the USB_DP or USB_DM signal routing, changes to the parametric trim fields may be useful.

62.7 USBPHY register descriptions

62.7.1 USBPHY memory map

USBHS1_PHY.USBPHY base address: 4010_A000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Power Down (PWD)	32	RW	001E_1C00h
4h	Power Down (PWD_SET)	32	RW	001E_1C00h
8h	Power Down (PWD_CLR)	32	RW	001E_1C00h
Ch	Power Down (PWD_TOG)	32	RW	001E_1C00h
10h	TX Control (TX)	32	RW	1007_0707h
14h	TX Control (TX_SET)	32	RW	1007_0707h
18h	TX Control (TX_CLR)	32	RW	1007_0707h
1Ch	TX Control (TX_TOG)	32	RW	1007_0707h
20h	RX Control (RX)	32	RW	0000_0000h
24h	RX Control (RX_SET)	32	RW	0000_0000h
28h	RX Control (RX_CLR)	32	RW	0000_0000h
2Ch	RX Control (RX_TOG)	32	RW	0000_0000h
30h	General Purpose Control (CTRL)	32	RW	C000_0000h
34h	General Purpose Control (CTRL_SET)	32	RW	C000_0000h
38h	General Purpose Control (CTRL_CLR)	32	RW	C000_0000h
3Ch	General Purpose Control (CTRL_TOG)	32	RW	C000_0000h
40h	Status (STATUS)	32	RW	See section
50h	Debug 0 (DEBUG0)	32	RW	7F18_0000h
54h	Debug 0 (DEBUG0_SET)	32	RW	7F18_0000h
58h	Debug 0 (DEBUG0_CLR)	32	RW	7F18_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
5Ch	Debug 0 (DEBUG0_TOG)	32	RW	7F18_0000h
80h	Version (VERSION)	32	R	0500_0000h
90h	IP Block (IP)	32	RW	0006_0000h
94h	IP Block (IP_SET)	32	RW	0006_0000h
98h	IP Block (IP_CLR)	32	RW	0006_0000h
9Ch	IP Block (IP_TOG)	32	RW	0006_0000h
A0h	PLL SIC (PLL_SIC)	32	RW	0091_2000h
A4h	PLL SIC (PLL_SIC_SET)	32	RW	0091_2000h
A8h	PLL SIC (PLL_SIC_CLR)	32	RW	0091_2000h
ACh	PLL SIC (PLL_SIC_TOG)	32	RW	0091_2000h
C0h	VBUS Detect (USB1_VBUS_DETECT)	32	RW	0070_0004h
C4h	VBUS Detect (USB1_VBUS_DETECT_SET)	32	RW	0070_0004h
C8h	VBUS Detect (USB1_VBUS_DETECT_CLR)	32	RW	0070_0004h
CCh	VBUS Detect (USB1_VBUS_DETECT_TOG)	32	RW	0070_0004h
D0h	VBUS Detect Status (USB1_VBUS_DET_STAT)	32	R	0000_0001h
D4h	VBUS Detect Status (USB1_VBUS_DET_STAT_SET)	32	R	0000_0001h
D8h	VBUS Detect Status (USB1_VBUS_DET_STAT_CLR)	32	R	0000_0001h
DCh	VBUS Detect Status (USB1_VBUS_DET_STAT_TOG)	32	R	0000_0001h
E0h	Charger Detect (USB1_CHRG_DETECT)	32	RW	8018_0000h
E4h	Charger Detect (USB1_CHRG_DETECT_SET)	32	RW	8018_0000h
E8h	Charger Detect (USB1_CHRG_DETECT_CLR)	32	RW	8018_0000h
ECh	Charger Detect (USB1_CHRG_DETECT_TOG)	32	RW	8018_0000h
F0h	Charger Detect Status (USB1_CHRG_DET_STAT)	32	R	0000_0000h
F4h	Charger Detect Status (USB1_CHRG_DET_STAT_SET)	32	R	0000_0000h
F8h	Charger Detect Status (USB1_CHRG_DET_STAT_CLR)	32	R	0000_0000h
FCh	Charger Detect Status (USB1_CHRG_DET_STAT_TOG)	32	R	0000_0000h
100h	Analog Control (ANACTRL)	32	RW	0200_0400h
104h	Analog Control (ANACTRL_SET)	32	RW	0200_0400h
108h	Analog Control (ANACTRL_CLR)	32	RW	0200_0400h
10Ch	Analog Control (ANACTRL_TOG)	32	RW	0200_0400h
130h	Trim (TRIM_OVERRIDE_EN)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
134h	Trim (TRIM_OVERRIDE_EN_SET)	32	RW	See section
138h	Trim (TRIM_OVERRIDE_EN_CLR)	32	RW	See section
13Ch	Trim (TRIM_OVERRIDE_EN_TOG)	32	RW	See section
140h	PFD A (PFDA)	32	RW	0101_0101h
144h	PFD A (PFDA_SET)	32	RW	0101_0101h
148h	PFD A (PFDA_CLR)	32	RW	0101_0101h
14Ch	PFD A (PFDA_TOG)	32	RW	0101_0101h

62.7.2 Power Down (PWD)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
PWD	0h	Power Down
PWD_SET	4h	Writing 1 to a bit in this register ensures that the corresponding bit in PWD is 1
PWD_CLR	8h	Writing 1 to a bit in this register ensures that the corresponding bit in PWD is 0
PWD_TOG	Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in PWD

Function

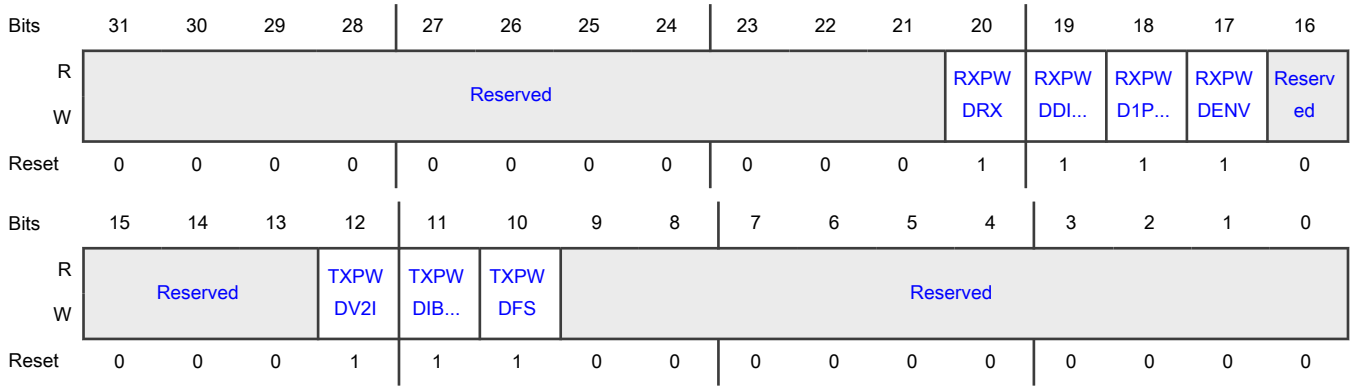
Provides overall control and status indication of the USBPHY power state.

The fields in this register have autopower-down writing of 1 and 0, depending on the configuration of fields in other registers. These features are intended to reduce the amount of software interaction needed when entering and leaving the Suspend bus state.

You must enable USBPHY's main reference bias generator separately, including the specific bias circuits required for the PLL and those controlled with this register. See PLL_SIC[REFBIAS_PWD] and PLL_SIC[REFBIAS_PWD_SEL] for information regarding how the main reference bias is controlled.

Before programming this register, you must enable the USBPHY clocks in USBPHY's General Purpose Control (CTRL) and PLL SIC (PLL_SIC).

Diagram



Fields

Field	Function
31-21 —	Reserved
20 RXPWDRX	Power Down USBPHY Receiver Circuits Disables the RX circuits except the FS differential comparator, such as the HS RX differential receiver, HS envelope detector, and HS host disconnect comparator, along with their RX current mirror bias circuits. The overall control of the RX current bias hardware module is shared with the battery charge circuit, but this field powers down the other aforementioned RX circuits unconditionally. If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_PHY_PWD] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state. 0b - Enable 1b - Disable or power down
19 RXPWDDIFF	Power Down USB HS Differential Receiver Disables the USB HS RX differential receiver that is used for data reception in HS operation. If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_PHY_PWD] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state. 0b - Enable 1b - Disable or power down
18 RXPWD1PT1	Power Down USB FS Differential Receiver Disables the USB FS RX differential receiver that is used for data reception in both FS and LS operations. This field does not affect the bias and operation of the FS differential receiver.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_PHY_PWD] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state. 0b - Enable 1b - Disable or power down
17 RXPWDENV	Power Down USB HS RX Envelope Detector Disables the USB HS RX envelope detector and host disconnect comparator. The USB HS RX envelope detector generates the squelch signal used to qualify the HS RX data. The host disconnect comparator is used in Host mode to check for a disconnection from the remote device. If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_PHY_PWD] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state. 0b - Enable 1b - Disable or power down
16-13 —	Reserved
12 TXPWDV2I	Power Down USBPHY TX V-I Converter and Current Mirror Disables the USBPHY TX V-I converter and current mirror. The V-I converter provides the currents used in the HS TX drivers and depends on both USBPHY's bias module and the bias current block. If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_PHY_PWD] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state. 0b - Enable 1b - Disable or power down
11 TXPWDIBIAS	Power Down USBPHY TX Current Bias Block Disables the USBPHY current bias block for the TX that provides mirrored currents to track USBPHY's bias module reference current generator for several functions in the TX, RX, and detector sections. Control of the current bias block is shared with the battery charge circuit. Writing 1 to this field does not power down the circuit unless the corresponding field in the battery charge control is also 1 for power down. The field must be 1 only when the USB is in the Suspend bus state. This effectively powers down the entire USB transmit path. If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_PHY_PWD] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enable 1b - Disable or power down
10 TXPWDFS	Power Down USB FS TX Drivers Disables the USB FS TX drivers that must have their bias enabled during USB data communication, including resume and bus reset signaling. They are used in HS, FS, and LS because the pulldown of the FS TX drivers provides local HS termination. If this field is 1, configure them to high-impedance state. If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_PHY_PWD] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state. 0b - Provide bias to enable 1b - Disable or power down
9-0 —	Reserved

62.7.3 TX Control (TX)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
TX	10h	TX Control
TX_SET	14h	Writing 1 to a bit in this register ensures that the corresponding bit in TX is 1
TX_CLR	18h	Writing 1 to a bit in this register ensures that the corresponding bit in TX is 0
TX_TOG	1Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in TX

Function

Allows adjustment of transmit circuit parameters.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			Reserved			Reserved	Reserved	Reserved		Reserved	Reserved	TXCAL45DP			
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		Reserved	Reserved	TXCAL45DN				Reserved				D_CAL			
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Fields

Field	Function
31-29 —	Reserved
28-26 —	The value of this reserved bit must remain 100b.
25 —	The value of this reserved bit must remain 0b.
24 —	The value of this reserved bit must remain 0b.
23-22 —	Reserved
21 —	The value of this reserved bit must remain 0b.
20 —	Reserved
19-16 TXCAL45DP	DP Series Termination Resistance Trim Specifies the trim value of the 45 Ω DP series termination resistance. Decode to trim the nominal 45 Ω series termination resistance to the USB_DP output pin. When TRIM_OVERRIDE_EN[TX_CAL45DP_OVERRIDE] is 1, the values in this field are used to trim the termination resistance. When TRIM_OVERRIDE_EN[TX_CAL45DP_OVERRIDE] is 0, the resistance trim values in TRIM_OVERRIDE_EN[USBPHY_TX_CAL45DP] are used instead. Maximum resistance is at value 0000b, minimum resistance is at value 1111b, and resistance is centered at value 0111b. For this USBPHY, each incrementally increasing trim value decreases the target resistance by about 3.5% compared to the next lower value.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Trimming this resistance impacts both the overshoot and undershoot of the FS TX output and the amplitude of the HS TX output.
15-14 —	Reserved
13 —	The value of this reserved bit must remain 0b.
12 —	Reserved
11-8 TXCAL45DN	DM Series Termination Resistance Trim Specifies the trim value of the 45 Ω DM series termination resistance. Decode to trim the nominal 45 Ω series termination resistance to the USB_DM output pin. When TRIM_OVERRIDE_EN[TX_CAL45DM_OVERRIDE] is 1, the values in this field are used to trim the termination resistance. When TRIM_OVERRIDE_EN[TX_CAL45DM_OVERRIDE] is 0, the resistance trim values in TRIM_OVERRIDE_EN[USBPHY_TX_CAL45DN] are used instead. Maximum resistance is at value 0000b, minimum resistance is at value 1111b, and resistance is centered at value 0111b. For this USBPHY, each incrementally increasing trim value decreases the target resistance by about 3.5% compared to the next lower value. Trimming this resistance impacts both the overshoot and undershoot of the FS TX output and the amplitude of the HS TX output.
7-4 —	Reserved
3-0 D_CAL	HS TX Output Current Trim Specifies the trim value of the HS TX output current. Decode to trim the nominal 17.78 mA current source for the HS TX drivers on USB_DP and USB_DM. This output current is directly proportional to the amplitude of the HS TX eye diagram. When TRIM_OVERRIDE_EN[TX_D_CAL_OVERRIDE] is 1, the values in this field are used to trim the output current. When TRIM_OVERRIDE_EN[TX_D_CAL_OVERRIDE] is 0, the current trim values in TRIM_OVERRIDE_EN[USBPHY_TX_D_CAL] are used instead. 0000b - Maximum current, approximately 19% above nominal 0111b - Nominal 1111b - Minimum current, approximately 19% below nominal

62.7.4 RX Control (RX)

Offset

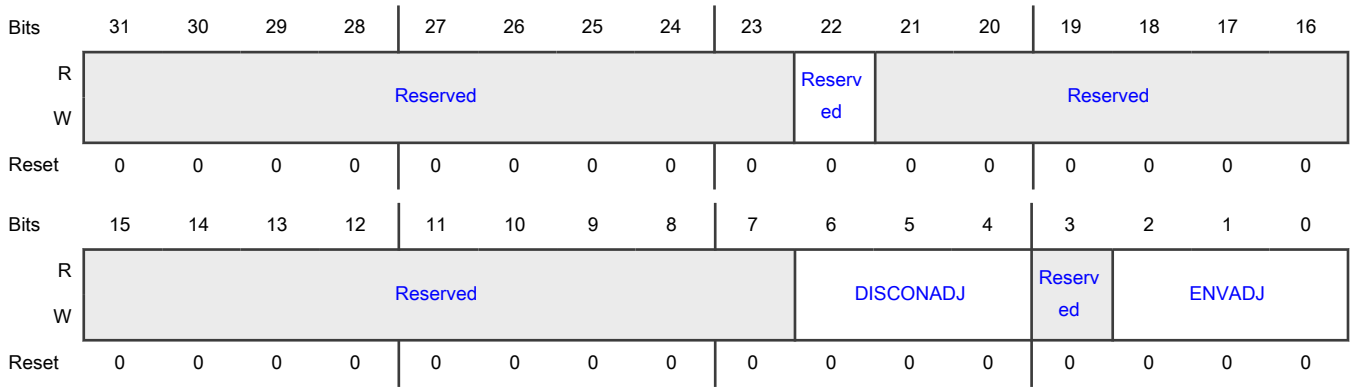
This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
RX	20h	RX Control
RX_SET	24h	Writing 1 to a bit in this register ensures that the corresponding bit in RX is 1
RX_CLR	28h	Writing 1 to a bit in this register ensures that the corresponding bit in RX is 0
RX_TOG	2Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in RX

Function

Handles receive path controls.

Diagram



Fields

Field	Function
31-23 —	Reserved
22 —	The value of this reserved bit must remain 0b.
21-7 —	Reserved
6-4 DISCONADJ	Disconnect Detector Trip Point Adjusts the trip point (trip-level voltage) for HS disconnect detector used in Host mode. 000b - 0.56875 V 001b - 0.55000 V

Table continues on the next page...

Table continued from the previous page...

Field	Function
	010b - 0.58125 V 011b - 0.60000 V 1xxb - Reserved
3 —	Reserved
2-0 ENVADJ	Envelope Detector Trip Point Adjusts the trip point (trip-level voltage) for the envelope detector used for HS squelch (see Squelch detector). The following values are nominal DC values to indicate the effect of changing this field. AC values measured during compliance testing are higher. 000b - 0.1000 V 001b - 0.1125 V 010b - 0.1250 V 011b - 0.0875 V 1xxb - Reserved

62.7.5 General Purpose Control (CTRL)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
CTRL	30h	General Purpose Control
CTRL_SET	34h	Writing 1 to a bit in this register ensures that the corresponding bit in CTRL is 1
CTRL_CLR	38h	Writing 1 to a bit in this register ensures that the corresponding bit in CTRL is 0
CTRL_TOG	3Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in CTRL

Function

Handles OTG and host controls.

This register also includes interrupt enables, connectivity detect enables, and results.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	UTMI_SU...	Reserved	OTG_ID_...	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	ENAUTOC...	ENAUTOC...	AUTORES...	WAKEUP_...	ENIRQWA...
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUTMIL...	ENUTMIL...	DATA_ON...	DEVPLUG...	ENIRQDE...	RESUME_...	ENIRQRE...	RESUMEI...	ENOTGID...	OTG_ID_...	DEVPLUG...	ENDEVPL...	HOSTDIS...	ENIRQHO...	ENHOSTD...	ENOTG_I...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SFTRST	Software Reset Soft-resets USBPHY's Power Down (PWD), TX Control (TX), RX Control (RX), General Purpose Control (CTRL), STATUS[OTGID_STATUS], Debug 0 (DEBUG0), and IP Block (IP), along with resetting state machines in USBPHY. Write 0 to this field to release USBPHY from reset. 0b - Release from reset 1b - Soft-reset
30 CLKGATE	UTMI Clock Gate Gates UTMI clocks. Write 1 to this field to save power when USB is not actively used. The configuration state of this field is retained when the clock is gated. If a wake-up event exists when USB is suspended and CTRL[ENAUTOCLR_CLKGATE] is 1, this field becomes 0 automatically. This field becomes 1 automatically at the falling edge of the signal from the USB controller that drives CTRL[UTMI_SUSPENDM] to enter the Suspend state. 0b - Run clocks 1b - Gate clocks
29 UTMI_SUSPENDM	UTMI Suspend Indicates a power-down state. If all power-down fields in USBPHY's Power Down (PWD) are 1, this field is forced to 0. Otherwise, this field reflects a synchronized version of the SuspendM signal from the USB controller to USBPHY. UTMI_SUSPENDM is negative logic because the UTMI specification requires it to be. 0b - Not suspended 1b - Suspended
28	The value of this reserved bit must remain 0b.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
27 OTG_ID_VALUE	OTG ID Value Indicates the results of the ID pin status for the USB cable plugged into the local micro- or mini-AB receptacle. If this field is 0, the ID resistance to ground is less than Ra_Plug_ID, indicating the host (A) side. If this field is 1, the ID resistance is greater than Rb_Plug_ID, indicating the device (B) side. You can use this field to reflect the status of USBPHY's internal USB_ID pin detector, or either of the local or external ID pin overrides depending on the configuration of USB1_VBUS_DETECT[ID_OVERRIDE_EN] and USB1_VBUS_DETECT[EXT_ID_OVERRIDE_EN]. The function of this field is very similar to that of STATUS[OTGID_STATUS], but this field has additional debounce and system clock synchronization logic to filter the glitches on the USB_ID pin. 0b - Host 1b - Device
26 —	This field has no function on this chip.
25 —	This field has no function on this chip.
24 —	The value of this reserved bit must remain 0b.
23 —	This field has no function on this chip.
22 —	This field has no function on this chip.
21 —	This field has no function on this chip.
20 ENAUTOCLR_PHY_PWD	PHY PWD Autoclear Enable Enables automatic writing of 0 to the fields in USBPHY's Power Down (PWD) if there is a wake-up event when USB is suspended. If wake-up is required automatically without software interaction, you must write 1 to this field. Recognizing the rising edge of the UTMI_SUSPENDM signal driven from the USB controller to USBPHY triggers the wake-up event for this feature. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 ENAUTOCLR_CLKGATE	Autoclear Clock Gate Enable Enables automatic writing of 0 to CLKGATE if there is a wake-up event when USB is suspended. If wake-up is required automatically without software interaction, you must write 1 to this field. Recognizing the rising edge of the UTMI_SUSPENDM signal driven from the USB controller to USBPHY triggers the wake-up event for this feature. 0b - Disable 1b - Enable
18 AUTORESUME_EN	Autoresume Enable Enables the autoresume feature. When this field is 1, USBPHY uses a 32 kHz clock to send resume signaling to respond to the device remote wake-up (for Host mode only). This feature is useful when the USB PLL is off and the reference clock is also powered down. 0b - Disable 1b - Enable
17 WAKEUP_IRQ	Wake-Up Interrupt Specifies that the USB controller has stopped driving the SuspendM signal in its active-low state to USBPHY. If ENIRQWAKEUP is 1, the state of this field is passed to USBPHY's combined interrupt output signal. Reset this field by writing 1 to the SCT clear address space, and not by a general write.
16 ENIRQWAKEUP	Wake-Up Interrupt Enable Enables WAKEUP_IRQ detection results to affect USBPHY's combined interrupt output signal. 0b - Disable 1b - Enable
15 ENUTMILEVEL3	UTMI Level 3 Enable Enables the UTMI+ level 3 operation for USBPHY. If an embedded host use case must support an external FS hub with an LS device connected, you must write 1 to this field. 0b - Disable 1b - Enable
14 ENUTMILEVEL2	UTMI Level 2 Enable Enables the UTMI+ level 2 operation for USBPHY. If an embedded host use case must support an LS device, you must write 1 to this field. 0b - Disable 1b - Enable
13	APB Clock Switch Option

Table continues on the next page...

Table continued from the previous page...

Field	Function
DATA_ON_LRADC	Specifies an APB clock switch option. This is one of the fields that affects muxing and the behavior of the APB clock used in USBPHY. On some chips, changes to the value of this field may be required to allow programming of registers during the USB Suspend state or for proper recognition of wake-up and disconnect. To avoid unexpected USB behavior, do not change the value of this field without a specific recommendation from NXP.
12 DEVPLUGIN_IRQ	Device Plug-In Interrupt Specifies the device connected indicator for nonstandard resistive plugged-in detection and that the USB cable attachment status between the device and host has changed. Only if ENDEVPLUGINDETECT is 1, the value of this field is valid. If ENIRQDEVPLUGIN is 1, the state of this field is passed to USBPHY's combined interrupt output signal. Reset this field by writing 1 to the SCT clear address space, and not by a general write.
11 ENIRQDEVPLUGIN	Enable Interrupt for Nonstandard Resistive Plugged-In Detection Enables the detection results in DEVPLUGIN_IRQ to affect USBPHY's combined interrupt output signal. 0b - Disable 1b - Enable
10 RESUME_IRQ	Resume Interrupt Specifies whether the host is sending a Resume signal after a Suspend signal. This field also becomes 1 after a bus reset during the Suspend state. Use this field to wake up from this state for either the resume or the reset case. If ENIRQRESUMEDETECT is 1, the state of this field is passed to USBPHY's combined interrupt output signal. Reset this field by writing 1 to the SCT clear address space or by a general register write. The value of RESUMEIRQSTICKY also affects the behavior of this field and its IRQ signal. 0b - No resume interrupt 1b - Resume interrupt
9 ENIRQRESUMEDETECT	Resume Detection Interrupt Enable Enables the detection results in RESUME_IRQ to affect USBPHY's combined interrupt output signal. You must write 1 to this field only after the device has entered the Suspend state. 0b - Disable 1b - Enable
8 RESUMEIRQSTICKY	Resume Interrupt Sticky Specifies until when RESUME_IRQ remains 1. 0b - During the resume or reset state signaling period

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Until you write 0 to it
7 ENOTGIDDETECT	Enable Internal OTG ID Detector Enables the circuit to detect the resistance of the USB_ID pin when monitoring the cable plugged into the micro- or mini-AB receptacle. On some chips, the USB controller provides an alternate method to enable the pullup for the USB_ID pin. Either of the two USB_ID pullup enable methods is used. To avoid extra power consumption, this field must remain 0 whenever USB_ID pin monitoring is not required. 0b - Disable 1b - Enable
6 OTG_ID_CHG_IRQ	OTG ID Change Interrupt Specifies whether the value of the OTG ID pin is changed. If ENOTG_ID_CHG_IRQ is 1, the state of this field is passed to USBPHY's combined interrupt output signal. Reset this field by writing 1 to the SCT clear address space, and not by a general write. 0b - No ID change interrupt 1b - ID change interrupt
5 DEVPLUGIN_POLARITY	Device Plug-In Polarity Specifies the device plug-in polarity interrupt configuration for nonstandard resistive plugged-in detection. For Device mode, if this field is 0, the interrupt is tripped when the USB cable from the host to the device is plugged in. If this field is 1, the interrupt is tripped when the USB cable from the host to the device is unplugged. 0b - Plugged in 1b - Unplugged
4 ENDEVPLUGINDETECT	Enable Nonstandard Resistive Plugged-In Detection Enables connection of nominal 200 kΩ pullup resistors to both the USB_DP and USB_DM pins as a method of detecting when a USB cable is attached in Device mode. This field must remain 0 for normal USB data communication or when using USBHSDCD for battery charger detection, according to *USB Battery Charger Specification Revision 1.2*, or any other detection mechanism for USB cable plug-in. When this field is 1, all methods for enabling the local 15 kΩ pulldown resistors, such as ANACTRL[DEV_PULLDOWN], must be 0. STATUS[DEVPLUGIN_STATUS] reports the results of this detection method. 0b - Disable 1b - Enable
3 HOSTDISCONDETECT_IRQ	Host Disconnect Detection Interrupt Specifies whether the host has detected USB device disconnection in HS mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If CTRL[HOSTDISCONDETECT_IRQ] is 1, the state of this field is passed to USBPHY's combined interrupt output signal. Reset this field by writing 1 to the SCT clear address space, and not by a general write. 0b - Connected 1b - Disconnected
2 ENIRQHOSTDISCON	Enable Interrupt for Host Disconnect Enables the detection results in HOSTDISCONDETECT_IRQ to affect USBPHY's combined interrupt output signal. This field becomes 1 in Host mode for HS operation after software recognizes that a USB device connection has occurred and after writing 1 to ENHOSTDISCONDETECT. 0b - Disable 1b - Enable
1 ENHOSTDISCONDETECT	Host Disconnect Detection Enable Enables HS disconnect detector for Host mode. This field must be 1 to allow the UTMI controller to observe the results of the HS disconnect detector, which is sampled at an appropriate time, after the host sends an SOF packet. After recognizing that an HS device is connected, you must write 1 to this field. The state of this detector is observed using STATUS[HOSTDISCONDETECT_STATUS]. 0b - Disable 1b - Enable
0 ENOTG_ID_CHG_IRQ	OTG ID Change Interrupt Enable Enables the detection results in OTG_ID_CHG_IRQ to affect USBPHY's combined interrupt output signal. 0b - Disable 1b - Enable

62.7.6 Status (STATUS)

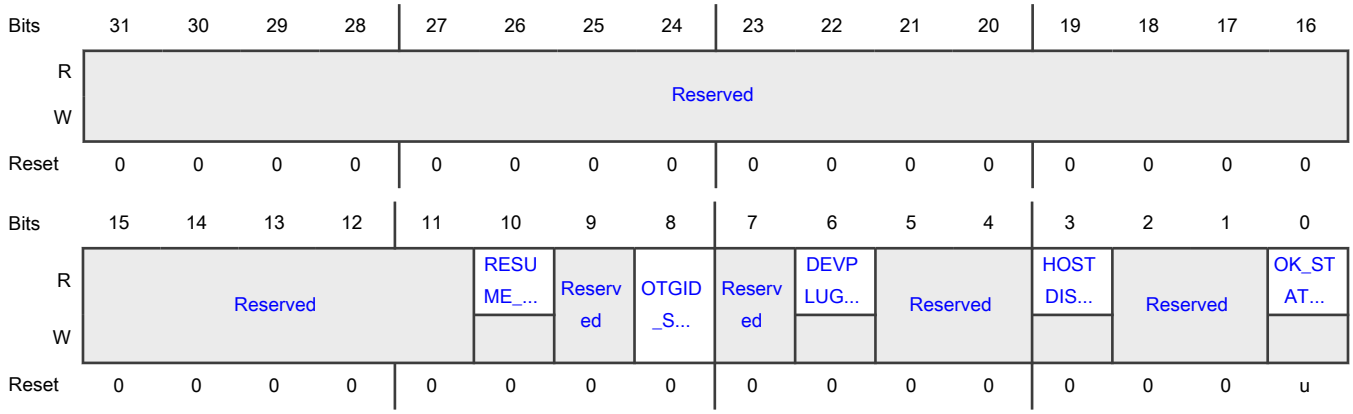
Offset

Register	Offset
STATUS	40h

Function

Holds results of IRQ and other detects.

Diagram



Fields

Field	Function
31-11 —	Reserved
10 RESUME_STATUS	Resume Status Indicates that the host is sending a wake-up request after the Suspend state and has triggered an interrupt. This field only provides a valid indication to resume when operating in FS or HS, after being in the Suspend bus state. The status field may toggle during USB data packet communication. Also, it does not indicate resume status when operating as a host that is directly attached to an LS device.
9 —	Reserved
8 OTGID_STATUS	OTG ID Status Specifies the results of the ID pin status for the USB cable plugged into the local micro- or mini-AB receptacle. The value of this field is also used for the IDDIG signal sent from USBPHY to the USB controller. You can use this field to reflect the status of USBPHY's internal USB_ID pin detector or either of the local or external ID pin overrides, depending on the configuration of USB1_VBUS_DETECT[ID_OVERRIDE_EN] and USB1_VBUS_DETECT[EXT_ID_OVERRIDE_EN]. If this field is 0, the ID resistance to ground is less than Ra_Plug_ID, indicating the host (A) side. If this field is 1, the ID resistance is greater than Rb_Plug_ID, indicating the device (B) side. 0b - Host 1b - Device
7 —	Reserved
6	Status Indicator for Nonstandard Resistive Plugged-In Detection

Table continues on the next page...

Table continued from the previous page...

Field	Function
DEVPLUGIN_STATUS	Indicates whether the device is connected on the USB_DP and USB_DM lines using the nonstandard resistive plugged-in detection method that CTRL[ENDEVPLUGINDETECT] controls. When a USB cable attached to a remote host is attached to the local device, the 15 kΩ host pulldowns override the high value resistors used in this detection method. 0b - No attachment detected 1b - Cable attachment detected
5-4 —	Reserved
3 HOSTDISCONDETECT_STATUS	Host Disconnect Status Indicates whether the USB cable disconnect is detected at the local host (downstream) port when in HS mode. 0b - Not detected 1b - Detected
2-1 —	Reserved
0 OK_STATUS_3V	USB 3.3 V and 1.8 V Supply Status Indicates the power status of USBPHY's noncore supply domains. The operation of this field depends on the state of ANACTRL[LVI_EN]. When ANACTRL[LVI_EN] is 1, this field is only 1 if both the USB 1.8 V and USB 3.3 V power rails are in their valid ranges. If this field is 0, one or both of the 1.8 V and 3.3 V supplies to USBPHY are not powered. 0b - Not powered 1b - Powered

62.7.7 Debug 0 (DEBUG0)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
DEBUG0	50h	Debug 0
DEBUG0_SET	54h	Writing 1 to a bit in this register ensures that the corresponding bit in DEBUG0 is 1
DEBUG0_CLR	58h	Writing 1 to a bit in this register ensures that the corresponding bit in DEBUG0 is 0

Table continues on the next page...

Table continued from the previous page...

Register	Offset	Description
DEBUG0_TOG	5Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in DEBUG0

Function

Controls debug features of USBPHY for testing and characterization. Some fields in this register also allow additional configurability for the 15 kΩ pulldown resistors and the behavior of the signals related to the OTG ID detection.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	Reserved	Reserved	Reserved					Reserved				Reserved			
Reset	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			Reserved	Reserved				Reserved		ENHSTPULLDOWN		HSTPULLDOWN		Reserved	OTGIDPI...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	The value of this reserved bit must remain 1b.
29 —	The value of this reserved bit must remain 1b.
28-25 —	The value of this reserved bit must remain 1111b.
24 —	The value of this reserved bit must remain 1b.
23-21 —	Reserved
20-16	The value of this reserved bit must remain 11000b.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
15-13 —	Reserved
12 —	The value of this reserved bit must remain 0b.
11-8 —	The value of this reserved bit must remain 0000b.
7-6 —	Reserved
5-4 ENHSTPULLDOWN	Enable Host Pulldown Overdrive Mode Enables Host Pulldown Overdrive mode. Write 1b to bit 5 to override the control of the USB_DP 15 kΩ pulldown, which the USB controller normally performs. Write 1b to bit 4 to override the control of the USB_DM 15 kΩ pulldown, which the USB controller normally performs. Write 00b to both bits to disable Host Pulldown Overdrive mode and return the control of the pulldown resistors to the USB controller. When in Host Pulldown Overdrive mode, HSTPULLDOWN further controls the connection of the individual pulldown resistors. Both pulldown resistors are also unconditionally enabled when ANACTRL[DEV_PULLDOWN] is 1, without considering the value of this field. 00b - Disable 01b - Enable
3-2 HSTPULLDOWN	Host Pulldown Overdrive Mode Selects whether to connect pulldown resistors on the USB_DP and USB_DM pins if the Host Pulldown Overdrive mode is enabled through ENHSTPULLDOWN. Write 1b to bit 3 to connect the 15 kΩ pulldown on the USB_DP line. Write 1b to bit 2 to connect the 15 kΩ pulldown on the USB_DM line. Write 00b to both bits to disconnect the resistors in Host Pulldown Override mode. 00b - Disconnect 01b - Connect
1 —	This field performs no function on this chip.
0	Hold OTG_ID

Table continues on the next page...

Table continued from the previous page...

Field	Function
OTGIDPIOLOCK	Holds the OTG ID value after the OTG_ID state from STATUS[OTGID_STATUS] is sampled. This provides power savings for the detection circuit that is used to determine the ID pin status by allowing the USB_ID pin detector to be disabled. The value of this field affects the value of STATUS[OTGID_STATUS] but does not affect the value of CTRL[OTG_ID_VALUE].

62.7.8 Version (VERSION)

Offset

Register	Offset
VERSION	80h

Function

Indicates the version of USBPHY.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STEP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 MAJOR	Major Indicates the major field of the RTL version.
23-16 MINOR	Minor Indicates the minor field of the RTL version.
15-0 STEP	Step Indicates the stepping of the RTL version.

62.7.9 IP Block (IP)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
IP	90h	IP Block
IP_SET	94h	Writing 1 to a bit in this register ensures that the corresponding bit in IP is 1
IP_CLR	98h	Writing 1 to a bit in this register ensures that the corresponding bit in IP is 0
IP_TOG	9Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in IP

Function

Contains patch and debug fields that modify digital operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							Reserved	Reserved			Reserved		Reserved	Reserved	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved													Reserved	Reserved	POWER_C...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 —	Reserved
24-23 —	The value of this reserved bit must remain 00b.
22-21 —	This field has no function on this chip.
20-19	The value of this reserved bit must remain 00b.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
18 —	The value of this reserved bit must remain 1b.
17 —	The value of this reserved bit must remain 1b.
16 —	The value of this reserved bit must remain 0b.
15-3 —	Reserved
2 —	The value of this reserved bit must remain 0b.
1 —	Do not change the value of this field without a specific recommendation from NXP.
0 POWER_CONTROL_SUSPEND_OPTION	Power Control Suspend Option Specifies the power control suspend option. This is one of the fields that affects muxing and the behavior of the APB clock used in USBPHY. On some chips, changes to the value of this field may be required to allow programming of registers during the USB Suspend state or for proper recognition of wake-up and disconnect. To avoid unexpected USB behavior, do not change the value of this field without a specific recommendation from NXP.

62.7.10 PLL SIC (PLL_SIC)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
PLL_SIC	A0h	PLL SIC
PLL_SIC_SET	A4h	Writing 1 to a bit in this register ensures that the corresponding bit in PLL_SIC is 1
PLL_SIC_CLR	A8h	Writing 1 to a bit in this register ensures that the corresponding bit in PLL_SIC is 0
PLL_SIC_TOG	ACh	Writing 1 to a bit in this register inverts the value of the corresponding bit in PLL_SIC

Function

Configures the operation of USBPHY's 480 MHz PLL and observes its lock status. Use this register to control USBPHY's main reference bias generator used for PLL, transceiver, and detector circuits. Additional fields in the register are used to configure power mode switching features.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PLL_LOCK	PLL_PRE...	Reserved			Reserved	Reserved		PLL_DIV_SEL		PLL_REG...	REFBIAS...	REFBIAS...	Reserved	Reserved	PLL_BYP...
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		PLL_ENA...	PLL_POW...	Reserved	Reserved	Reserved	Reserved	Reserved	PLL_EN_...	MISC2_C...	Reserved			Reserved	Reserved
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PLL_LOCK	USB PLL Lock Status Indicator Indicates whether the USB PLL is locked. This field becomes 1 after some delay when the PLL is powered up. USBPHY clears this field during the change in PLL software configuration settings, including PLL_POWER and PLL_DIV_SEL. If the PLL is configured for autopowerdown and USBPHY enters the Suspend bus state, this field becomes 0. 0b - Not locked 1b - Locked
30 PLL_PREDIV	PLL Pre-Divider Pre-divide selector between /1 or /2 to expand the range of the USB PLL input clock. USBPHY's PLL has a configurable internal pre-divider for its reference clock input. The pre-divider mux either passes the reference clock input directly to the PLL or uses a divide-by-two circuit between the module's reference clock input and the PLL loop. This feature allows support of 38.4, 40, 48, 60, or 64 MHz system reference clocks with the USB PLL. 0b - Uses the undivided reference clock for PLL loop 1b - Divides the reference clock by two for PLL loop
29-27 —	The value of this reserved bit must remain 000b.
26 —	The value of this reserved bit must remain 0b.

Table continues on the next page...

Table continued from the previous page...

Field	Function
25 —	The value of this reserved bit must remain 0b.
24-22 PLL_DIV_SEL	PLL Divider Value Configuration Controls the USB PLL feedback loop divider and allows use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator. USBPHY's PLL produces a 480 MHz output clock. When TRIM_OVERRIDE_EN[DIV_SEL_OVERRIDE] is 1, use the values in this field to control the PLL divider. When TRIM_OVERRIDE_EN[DIV_SEL_OVERRIDE] is 0, the PLL divider values in TRIM_OVERRIDE_EN[PLL_CTRL0_DIV_SEL] are used instead. If this field is 111b, it may give marginal jitter results. The input clock frequency used for the PLL reference clock input is the output of the pre-divider stage controlled by PLL_SIC[PLL_PREDIV] bit field. For most SOCs the pre-divider would not be enabled but this feature might be used if an SOC had a very high frequency but still stable system reference clock. 000b - Configure for a 32 MHz input clock (divide by 15) 001b - Configure for a 30 MHz input clock (divide by 16) 010b - Configure for a 24 MHz input clock (divide by 20) 011b - Reserved, not usable for USB operation (divide by 22) 100b - Configure for a 20 MHz input clock (divide by 24) 101b - Configure for a 19.2 MHz input clock (divide by 25) 110b - Configure for a 16 MHz input clock (divide by 30) 111b - Configure for a 12 MHz input clock (divide by 40)
21 PLL_REG_ENABLE	Enable PLL Regulator Enables the USB PLL regulator. The USB PLL operates in a voltage domain powered by a local regulator within USBPHY. Use this field to control the PLL regulator. The USB software stack must write 1 to this field. Then, at least 15 µs later, write 1 to PLL_POWER to avoid glitches on the PLL output clocks. You must power up USBPHY's main reference bias generator for the PLL regulator to operate. See REFBIAS_PWD and REFBIAS_PWD_SEL for information on how the main reference bias is controlled. 0b - Disable 1b - Enable
20 REFBIAS_PWD	Power Down Reference Bias Disables or powers down the main reference bias in direct power control mode. You must enable USBPHY's main reference bias circuits for the other bias circuits in USBPHY to operate, including the bias circuits required for the PLL and those controlled with Power Down (PWD).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field allows direct enable of the main reference bias circuits, independent of whether the USBPHY is in the Suspend bus state, but is only effective when REFBIAS_PWD_SEL is 1. The USB software stack must write 0 to this field together with writing 1 to REFBIAS_PWD_SEL to enable the needed bias circuits when performing battery charger detection or when using the VBUS_VALID comparator, if USBPHY is in the Suspend bus state with the USB PLL powered down. 0b - Enable 1b - Disable or power down
19 REFBIAS_PWD_SEL	Reference Bias Power Control Determines the method used to enable and disable USBPHY's main reference bias circuits. You must enable the main reference bias circuits for the other bias circuits in USBPHY to operate, including the bias circuits required for the PLL and those controlled using Power Down (PWD). The USB software stack must write 1 to this field together with writing 0 to REFBIAS_PWD to allow direct enable for the needed bias circuits when performing battery charger detection or when using the VBUS_VALID comparator, if USBPHY is in the Suspend bus state with the USB PLL powered down. For CLN40ULP PHY, it is recommended to write 1 to this field together with writing 0 to REFBIAS_PWD and writing 1 to MISC2_CONTROL0 to allow automatic power control when USBPHY is entering and leaving the Suspend bus state. 0b - PLL_POWER internal state signal 1b - REFBIAS_PWD
18 —	This field performs no function on this chip.
17 —	The value of this reserved bit must remain 0b.
16 PLL_BYPASS	Bypass USB PLL Controls the clock mux for the PLL main output used in USBPHY. The output clock signal is selected either from the 480 MHz output required for USB data communication or the PLL's reference input clock. Because the default state of this field is 1, you must write 0 to this field during the USB initialization routine. 0b - 480 MHz output clock 1b - Input reference clock
15-14 —	This field performs no function on this chip.
13 PLL_ENABLE	PLL Output Clock Enable Enables the main 480 MHz single-phase clock output from the USB PLL to the UTMI logic. This field works with MISC2_CONTROL0 to perform a PLL operation. You must power up the USB PLL using PLL_POWER first, before the clock output is enabled using this field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
12 PLL_POWER	USB PLL Powerup Control Allows the 480 MHz USB PLL to be powered up. This field, when 0, forces powerdown of the USB PLL and safestates all of its output clocks. USBPHY controls the internal PLL power-up state signal. When appropriately configured with other register fields, USBPHY can power down the PLL when USBPHY is in the Suspend bus state and the PLL clocks are not needed, and restart the PLL when exiting Suspend. In this Auto-Powerup or Auto-Powerdown mode, the USB software stack must write 1 to PLL_POWER when initializing the PLL and USBPHY. The value of this field reflects what you write to the register and it does not toggle as the autopowerup or autopowerdown conditions change. See REFBIAS_PWD_SEL, REFBIAS_PWD, PLL_REG_ENABLE, and MISC2_CONTROL0 to understand their influence on the USB PLL behavior and operation of its bias circuits. 0b - Power down 1b - Allow powerup
11 —	The value of this reserved bit must remain 0b.
10 —	The value of this reserved bit must remain 0b.
9 —	The value of this reserved bit must remain 0b.
8 —	The value of this reserved bit must remain 0b.
7 —	The value of this reserved bit must remain 0b.
6 PLL_EN_USB_CLKS	PLL Multi-Phase Clock Outputs Enable Enables the 480 MHz multiple-phase clock outputs from the USB PLL to the UTMI logic and HS RX data recovery circuits. This field works with MISC2_CONTROL0 to perform a PLL operation. You must power up the USB PLL by using PLL_POWER first, before the clock outputs are enabled using this field. 0b - Disable 1b - Enable
5	Miscellaneous Control

Table continues on the next page...

Table continued from the previous page...

Field	Function
MISC2_CONTROL0	Determines USB PLL operation status during the Suspend state. This field, together with PLL_POWER, determines when USBPHY's 480 MHz PLL is powered up. When this field is 0, the PLL is powered up depending on whether PLL_POWER is 1. When MISC2_CONTROL0 is 1, the PLL is powered up if PLL_SIC[PLL_POWER] is 1 and the value of CTRL[UTMI_SUSPENDM] from the USB controller is 1, but PLL is powered down when in the Suspend bus state, if CTRL[UTMI_SUSPENDM] is 0. For the CLN40ULP PHY, it is recommended to write 1 to this field together with writing 0 to REFBIAS_PWD and writing 1 to REFBIAS_PWD_SEL to allow automatic power control when USBPHY is entering and leaving the Suspend bus state. 0b - Power up PLL 1b - Power down PLL
4-2 —	The value of this reserved bit must remain 000b.
1 —	The value of this reserved bit must remain 0b.
0 —	The value of this reserved bit must remain 0b.

62.7.11 VBUS Detect (USB1_VBUS_DETECT)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
USB1_VBUS_DETECT	C0h	VBUS Detect
USB1_VBUS_DETECT_SET	C4h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_VBUS_DETECT is 1
USB1_VBUS_DETECT_CLR	C8h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_VBUS_DETECT is 0
USB1_VBUS_DETECT_TOG	CCh	Writing 1 to a bit in this register inverts the value of the corresponding bit in USB1_VBUS_DETECT

Function

Defines controls for several methods of USB VBUS detection and some additional out-of-band signaling functions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	Reserved			Reserved	DISCHAR...	Reserved			VBUSVALID_PWRUP_CMPS			Reserved	VBUSVAL...	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	EXT_VBU...	EXT_ID_...	ID_OVER...	ID_OVER...	VBUS_SOURCE_SEL		VBUSVAL...	VBUSVAL...	AVALID_...	BVALID_...	SESSEND...	VBUS_OV...	VBUSVALID_THRESH		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31 —	Reserved
30-28 —	Reserved
27 —	Reserved
26 DISCHARGE_VBUS	VBUS Discharge Resistor Enables (controls) a nominal 22 kΩ resistor between the USB1_VBUS pin and ground. You can use this field to accelerate the fall of the VBUS signal at the end of a session. 0b - Disable 1b - Enable
25-23 —	Reserved
22-20 VBUSVALID_PWRUP_CMPS	VBUS_VALID Comparator Enable Enables (powerups) the VBUS detection comparators. There are three different VBUS detection comparators or detectors included in USBPHY. This field allows any combination of those circuits to be enabled or powered down. Usually, not all of the VBUS detection methods are needed for a particular use case. The USB software stack must reset the individual fields to 0 for any VBUS detection methods that are not used to save power. To understand how to configure the VBUS detection comparators and observe their results, see the other fields in VBUS Detect (USB1_VBUS_DETECT) and VBUS Detect Status (USB1_VBUS_DET_STAT). Bit 20 of this field controls the VBUS_VALID comparator that is often used in Host mode. Bit 21 of this field controls the session valid detector that is often used in Device mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Bit 22 of this field controls the VBUS_VALID_3V detector that you can use in Host mode if a nonstandard VBUS voltage signal is connected to the USB1_VBUS pin. 0xxb - Disable or power down the VBUS_VALID_3V detector 1xxb - Enable the VBUS_VALID_3V detector x0xb - Disable or power down the session valid detector x1xb - Enable the session valid detector xx0b - Disable or power down the VBUS_VALID comparator xx1b - Enable the VBUS_VALID comparator
19 — 	Reserved
18 VBUSVALID_TO_B	VBUS_VALID Comparator Selection Selects the comparator used for VBUS_VALID. This field helps you select, between the VBUS_VALID comparator and session valid detector, the comparator used to report the VBUS_VALID results in USB1_VBUS_DET_STAT[VBUS_VALID]. The VBUS_VALID comparator is the most accurate and has a programmable threshold that VBUSVALID_THRESH defines. The session valid detector may be useful in systems using nonstandard VBUS voltages. The mux selection using this field happens before any VBUS_VALID selection controlled by VBUS_SOURCE_SEL and VBUSVALID_SEL, and it has an impact only if the VBUS overrides are not enabled and USB1_VBUS_DETECT[VBUSVALID_SEL] is 0. If this field is 0, use the VBUS_VALID comparator for the VBUS_VALID results. If this field is 1, use the session valid detector for the VBUS_VALID results. The session valid threshold is ≥ 0.8 V and ≤ 4.0 V. 0b - VBUS_VALID comparator 1b - Session valid detector
17-15 — 	Reserved
14 EXT_VBUS_OVERRIDE_EN	External VBUS Override Enable Enables the external VBUS override using a value supplied from outside USBPHY. USBPHY has an input to allow VBUS detection status to be supplied from an external digital signal in use cases where the USBPHY's internal detection method or software local overrides are not used. The connection of this external VBUS signal is chip-specific. When this field is 0, the VBUS status signals reported in USB1_VBUS_DET_STAT[VBUS_VALID], USB1_VBUS_DET_STAT[AVALID], USB1_VBUS_DET_STAT[BVALID], and USB1_VBUS_DET_STAT[SESSEND] and the VBUS_VALID,

Table continues on the next page...

Table continued from the previous page...

Field	Function
	AVALID, BVALID, and SESSEND signals are sent to the USB controller. The VBUS detection method configuration determines this, as described for other fields in this register, including VBUS_OVERRIDE_EN. When this field is 1, the VBUS status signals reported in USB1_VBUS_DET_STAT[VBUS_VALID], USB1_VBUS_DET_STAT[AVALID], and USB1_VBUS_DET_STAT[BVALID] and the VBUS_VALID, AVALID, and BVALID signals sent to the USB controller reflect the value on the external VBUS override signal. The VBUS status signal reported in USB1_VBUS_DET_STAT[SESSEND] and the SESSEND signal sent to the USB controller reflect the inverse of the value on the external VBUS override signal. 0b - Internal detector or local override 1b - External VBUS_VALID value
13 EXT_ID_OVERRIDE_EN	External ID Override Enable Enables the external ID pin status override using a value supplied from outside USBPHY. USBPHY has an input to allow OTG ID pin detection status to be supplied from an external digital signal in use cases where the USBPHY's internal detector methods or software local override are not used. The connection of this external ID pin status signal is chip-specific. When this field is 0, the ID pin status signal reported in STATUS[OTGID_STATUS] and the IDDIG signal sent to the USB controller are determined by the USBPHY's ID pin detection method or the local override value that ID_OVERRIDE and ID_OVERRIDE_EN configure. When this field is 1, the ID pin status signal reported in the following fields, along with the IDDIG signal sent to the USB controller, reflect the value of the external ID pin signal: • STATUS[OTGID_STATUS] • USB1_VBUS_DET_STAT[EXT_ID] 0b - Internal detector or local override 1b - External ID signal value
12 ID_OVERRIDE	ID Pin Status Local Override Specifies the ID pin status local override value. If the USB use case does not use USBPHY's dedicated USB_ID detection pin, this field provides an alternate method to reflect the ID pin status. If ID_OVERRIDE_EN is 1 and EXT_ID_OVERRIDE_EN is 0, then the value written in this field is used for the IDDIG signal and also reported in STATUS[OTGID_STATUS]. This field has no impact if either ID_OVERRIDE_EN or EXT_ID_OVERRIDE_EN is 0. See STATUS[OTGID_STATUS] to understand the state assignment convention of the ID pin value.
11 ID_OVERRIDE_EN	Enable Local ID Pin Status Override Enables (allows) local ID pin status override. If the USB use case does not use USBPHY's dedicated USB_ID detection pin, this field enables an alternate method to reflect the ID pin status. When this field is 1, the value of ID_OVERRIDE is used for the reported ID pin status. This field has an impact only if EXT_ID_OVERRIDE_EN is 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Use ID pin detector or external override 1b - Allow local override of ID pin detection status
10-9 VBUS_SOURCE_SEL	VBUS_VALID Source Selection Selects the source of the VBUS_VALID signal reported to the USB subsystem. If this field is 0, use the VBUS_VALID comparator results for the signal reported to the USB subsystem. If this field is 1 or 10, use the session valid comparator results for the signal reported to the USB subsystem. This is one of the fields that selects the source of the VBUS_VALID signal output from USBPHY for the rest of the USB subsystem. The way the output signal is connected to other portions of the USB subsystem, including the USB controller, is chip-specific. The VBUS_VALID source selections in this field takes effect only if VBUSVALID_SEL, VBUS_OVERRIDE_EN, and EXT_VBUS_OVERRIDE_EN are 0. This field does not impact the VBUS_VALID value reported in USB1_VBUS_DET_STAT[VBUS_VALID]. 00b - VBUS_VALID comparator result 01b - Session valid comparator result 10b - Session valid comparator result 11b - Reserved
8 VBUSVALID_SEL	VBUS_VALID Selection Selects the source of the VBUS_VALID signal reported to the USB subsystem. If this field is 0, use the VBUS_VALID comparator results for the signal reported to the USB subsystem. If this field is 1, use the VBUS_VALID_3V comparator results for the signal reported to the USB subsystem. This is one of the fields that selects the source of the VBUS_VALID signal output from USBPHY for the rest of the USB subsystem. The way the output signal is connected to other portions of the USB subsystem, including the USB controller is, chip-specific. The VBUS_VALID source selection in this field takes effect only if VBUS_OVERRIDE_EN and EXT_VBUS_OVERRIDE_EN are 0. This field does not impact the VBUS_VALID value reported in USB1_VBUS_DET_STAT[VBUS_VALID]. 0b - VBUS_VALID comparator result 1b - VBUS_VALID_3V comparator result
7 VBUSVALID_OVERRIDE	Override Value for the VBUS_VALID Signal Provides a value for the VBUS_VALID signal output from USBPHY to the rest of the USB subsystem if VBUS_OVERRIDE_EN is 1 and EXT_VBUS_OVERRIDE_EN is 0. The way the output signal is connected to other portions of the USB subsystem, including the USB controller, is chip-specific. The value of this field does not affect the VBUS_VALID value reported in USB1_VBUS_DET_STAT[VBUS_VALID].
6	Override Value for A-Device Session Valid

Table continues on the next page...

Table continued from the previous page...

Field	Function
AVALID_OVERRIDE	Provides the value for USB1_VBUS_DET_STAT[AVALID] and the UTMI AVALID signal output from USBPHY to the USB controller if VBUS_OVERRIDE_EN is 1 and EXT_VBUS_OVERRIDE_EN is 0.
5 BVALID_OVERRIDE	Override Value for B-Device Session Valid Provides the value for USB1_VBUS_DET_STAT[BVALID] and the UTMI BVALID signal output from USBPHY to the USB controller if VBUS_OVERRIDE_EN is 1 and EXT_VBUS_OVERRIDE_EN is 0.
4 SESSEND_OVERRIDE	Override Value for SESSEND Provides the value for USB1_VBUS_DET_STAT[SESSEND] and the UTMI SESSEND signal output from USBPHY to the USB controller if VBUS_OVERRIDE_EN is 1 and EXT_VBUS_OVERRIDE_EN is 0.
3 VBUS_OVERRIDE_EN	VBUS Detect Signal Local Override Enable Allows you to override the results from the VBUS_VALID and session valid comparators using the values in VBUSVALID_OVERRIDE, AVALID_OVERRIDE, BVALID_OVERRIDE, and SESSEND_OVERRIDE. This field has an impact only when EXT_VBUS_OVERRIDE_EN is 0. If this field is 0, use the results of the internal VBUS_VALID and session valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND. If this field is 1, use the override values for VBUS_VALID, AVALID, BVALID, and SESSEND. The VBUS_VALID, AVALID, BVALID, and SESSEND signals sent from USBPHY to the USB controller are affected by these bit selections. The values reported for VBUSVALID_OVERRIDE, AVALID_OVERRIDE, BVALID_OVERRIDE, and SESSEND_OVERRIDE are also affected, but the value of USB1_VBUS_DET_STAT[VBUS_VALID_3V] is not affected. This override method may be useful if VBUS detection is not done by using USBPHY's internal VBUS_VALID or session valid comparators. 0b - Results of VBUS_VALID and session valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND 1b - Override values for VBUS_VALID, AVALID, BVALID, and SESSEND
2-0 VBUSVALID_THRESH	VBUS Comparator Threshold Sets the threshold for the VBUS_VALID comparator, which is the most accurate method to determine the presence of a 5 V VBUS connection and includes hysteresis to minimize the need for software debounce of the detection. This comparator has ~50 mV of hysteresis to prevent chattering at the comparator trip point. 000b - 4.0 V 001b - 4.1 V 010b - 4.2 V 011b - 4.3 V 100b - 4.4 V 101b - 4.5 V 110b - 4.6 V

Table continues on the next page...

Table continued from the previous page...

Field	Function
	111b - 4.7 V

62.7.12 VBUS Detect Status (USB1_VBUS_DET_STAT)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

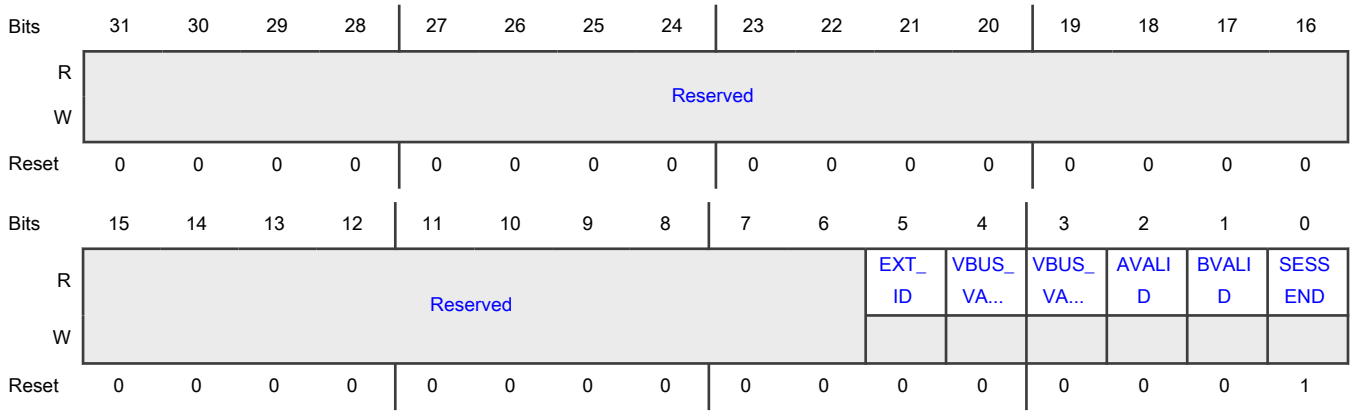
Register	Offset	Description
USB1_VBUS_DET_STAT	D0h	VBUS Detect Status
USB1_VBUS_DET_STAT_SET	D4h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_VBUS_DET_STAT is 1
USB1_VBUS_DET_STAT_CLR	D8h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_VBUS_DET_STAT is 0
USB1_VBUS_DET_STAT_TOG	DCh	Writing 1 to a bit in this register inverts the value of the corresponding bit in USB1_VBUS_DET_STAT

Function

Allows observation of status for several different types of USB VBUS detect functions.

This register also has a field to observe the external override signal for the OTG ID pin status.

Diagram



Fields

Field	Function
31-6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 EXT_ID	OTG ID External Override Status Reflects the value of the ID pin status for the external override input to USBPHY. This value is only valid when USB1_VBUS_DETECT[EXT_ID_OVERRIDE_EN] is 1. Because the function of this field is limited, it may be more useful to observe the ID pin status using CTRL[OTG_ID_VALUE] or STATUS[OTGID_STATUS] instead. To understand the state assignment convention of the ID pin value, see STATUS[OTGID_STATUS].
4 VBUS_VALID_3V	VBUS_VALID_3V Detector Status Reflects the output of USBPHY's VBUS_VALID_3V detector when USB1_VBUS_DETECT[VBUSVALID_PWRUP_CMPS] enables that detector. You cannot override this value. If this field is 0, the VBUS voltage is below the VBUS_VALID_3V threshold. If this field is 1, the VBUS voltage is above the VBUS_VALID_3V threshold. The internal VBUS_VALID_3V detector in USBPHY has a lower threshold for the voltage on the USB1_VBUS pin than either the session valid detector or the VBUS_VALID comparator. This signal may be useful for applications that provide a nonstandard VBUS voltage to USBPHY. 0b - Below threshold 1b - Above threshold
3 VBUS_VALID	VBUS Voltage Status Indicates the result of USBPHY's internal VBUS_VALID detection for the USB1_VBUS pin or external VBUS_VALID override. If this field is 0, the VBUS voltage is below the comparator threshold. If this field is 1, the VBUS voltage is above the comparator threshold. USB1_VBUS_DETECT[VBUSVALID_TO_B] selects the VBUS_VALID comparator used. The values of USB1_VBUS_DETECT[VBUSVALID_OVERRIDE] and USB1_VBUS_DETECT[VBUS_OVERRIDE_EN] do not affect the VBUS_VALID status reported, but you can overwrite it if USB1_VBUS_DETECT[EXT_VBUS_OVERRIDE_EN] is 1. The detection method used for VBUS_VALID in this status field depends on the value of USB1_VBUS_DETECT[VBUSVALID_TO_B]. 0b - Below threshold 1b - Above threshold
2 AVALID	A-Device Session Valid Status Reflects the output of the session valid detector in USBPHY when USB1_VBUS_DETECT[VBUSVALID_PWRUP_CMPS] enables that detector and you cannot override the value. If this field is 0, the VBUS voltage is below the session valid threshold.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If this field is 1, the VBUS voltage is above the session valid threshold. For local overrides, you can overwrite this field using USB1_VBUS_DETECT[VBUS_OVERRIDE_EN] and USB1_VBUS_DETECT[AVALID_OVERRIDE]. Besides, for external overrides, overwrite this field using USB1_VBUS_DETECT[EXT_VBUS_OVERRIDE_EN] and the external VBUS_VALID override input signal to USBPHY. 0b - Below threshold 1b - Above threshold
1 BVALID	B-Device Session Valid Status Reflects the output of the session valid detector in USBPHY when USB1_VBUS_DETECT[VBUSVALID_PWRUP_CMPS] enables that detector and you cannot override the value. If this field is 0, the VBUS voltage is below the session valid threshold. If this field is 1, the VBUS voltage is above the session valid threshold. For local overrides, you can overwrite this field using USB1_VBUS_DETECT[VBUS_OVERRIDE_EN] and USB1_VBUS_DETECT[BVALID_OVERRIDE]. Besides, for external overrides, overwrite this field using USB1_VBUS_DETECT[EXT_VBUS_OVERRIDE_EN] and the external VBUS_VALID override input signal to USBPHY. 0b - Below threshold 1b - Above threshold
0 SESSEND	Session End Indicator Reflects the inverse of the output of the session valid detector in USBPHY when USB1_VBUS_DETECT[VBUSVALID_PWRUP_CMPS] enables that detector and you cannot override the value. If this field is 0, the VBUS voltage is below the session valid threshold. If this field is 1, the VBUS voltage is above the session valid threshold. For local overrides, you can overwrite this field using USB1_VBUS_DETECT[VBUS_OVERRIDE_EN] and USB1_VBUS_DETECT[SESSEND_OVERRIDE]. Besides, for external overrides, overwrite this field using USB1_VBUS_DETECT[EXT_VBUS_OVERRIDE_EN] and the external VBUS_VALID override input signal to USBPHY. 0b - Above threshold 1b - Below threshold

62.7.13 Charger Detect (USB1_CHRG_DETECT)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
USB1_CHRG_DETECT	E0h	Charger Detect
USB1_CHRG_DETECT_SET	E4h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_CHRG_DETECT is 1
USB1_CHRG_DETECT_CLR	E8h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_CHRG_DETECT is 0
USB1_CHRG_DETECT_TOG	ECh	Writing 1 to a bit in this register inverts the value of the corresponding bit in USB1_CHRG_DETECT

Function

Enables USB battery charging (BC) v1.2 functions, including device mode charger detection and host mode charging port advertisement. You can control several of the USBPHY analog circuits for BC v1.2 functions by either using this register or the embedded USBHSDCD module, which has autosequencing state machines and a separate register space.

You must first enable USBPHY's main reference bias circuits for BC v1.2 functions by using either of the control methods (see MISC[REFBIAS_PWD] and MISC[REFBIAS_PWD_SEL]).

See the "USB High-Speed Device Charger Detection (USBHSDCD)" chapter for USBHSDCD module operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DCDSEL	Reserved							Reserved	Reserved		EN_B	CHK_CHR...	CHK_CON...	Reserved	
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved											VDM_SRC...	Reserved	PULLUP_...	DETECT_...	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 DCDSEL	DCD Selection Selects the control source for battery charging detection and advertisement circuits. If this field is 0, Charger Detect (USB1_CHRG_DETECT) controls the BC 1.2 functionality. If this field is 1, fields and state machines in the USBHSDCD module control the BC 1.2 functionality. You can control several of the USBPHY analog circuits associated with the USB BC v1.2 functions by either using control fields in this register or by using registers in the USBHSDCD module. Use of USBHSDCD is recommended for software simplification, for device mode charger detection because USBHSDCD can autosequence through the detection states. The USBHSDCD module is also recommended if charging port host advertisement is needed with this USBPHY.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Fields in USB1_CHRG_DETECT 1b - Fields and state machines in the USBHSDCD module
30-24 —	Reserved
23 —	Do not change the value of this field without a specific recommendation from NXP.
22-21 —	Reserved
20 EN_B	Selection of BC v1.2 Function Enable Enables the selection of the BC v1.2 functions depending on other fields. If this field is 0, the BC v1.2 functions, when controlled by this register, are enabled. If this field is 1, the BC v1.2 functions, when controlled by this register, are disabled. This field serves as a qualifier for CHK_CHRG_B, CHK_CONTACT, and DETECT_SEC when DCDSEL is 0. You must disable the charger detection operation prior to the start of a normal USB data communication. 0b - Enable 1b - Disable
19 CHK_CHRG_B	BC Charger Detection Function Enable Enables the BC v1.2 primary detection function for charger detect depending on other fields. If this field is 0, the BC charger detection functions, when controlled by this register, are enabled. If this field is 1, the BC charger detection functions, when controlled by this register, are disabled. A device uses primary detection to distinguish whether it is connected to a standard downstream port or a charging port. This field has an impact only when both DCDSEL and EN_B are 0. The results of this detection are observed in USB1_CHRG_DET_STAT[CHRG_DETECTED]. 0b - Enable 1b - Disable
18 CHK_CONTACT	BC Data Contact Detect Function Enable Enables the BC v1.2 data contact detect function for charger detect depending on other fields. If this field is 0, the BC data contact detect functions, when controlled by this register, are disabled. If this field is 1, the BC data contact detect functions, when controlled by this register, are enabled. A device uses the BC v1.2 method of data contact detection to discover attachment to a host port or dedicated charging port.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field has an impact only when both DCDSEL and EN_B are 0. The results of this detection are observed in USB1_CHRG_DET_STAT[PLUG_CONTACT]. 0b - Disable 1b - Enable
17-5 —	Reserved
4 VDM_SRC_ENABLE	VDM_SRC Function Enable Enables the BC v1.2 charging downstream port (CDP) advertisement signaling for charger detect depending on other fields. If this field is 0, the CDP advertisement signaling VDM_SRC functions, when controlled by this register, are disabled. If this field is 1, the CDP advertisement signaling VDM_SRC functions, when controlled by this register, are enabled. You must only use this signaling in embedded host mode according to the BC v1.2 specification. This field only affects when DCDSEL, EN_B, and DETECT_SEC are 0, and CHK_CHRG_B is 1. 0b - Disable 1b - Enable
3 —	The value of this reserved bit must remain 0b.
2 PULLUP_DP	DP Pullup Resistor Enable Override Control Enables the pullup resistor on the USB_DP pin required for FS Device mode operation, which is normally controlled using the UTMI bus signals sent to USBPHY from the USB controller. If this field is 0, the DP pullup resistor is controlled only with UTMI bus signals. If this field is 1, the DP pullup resistor is force enabled. If this field is 1, the pullup resistor is enabled independently of the UTMI signals to USBPHY. The USB software stack uses this capability during charger type detection, according to the obsolete *USB Battery Charging Specification, Revision 1.1*. For *USB Battery Charging Specification, Revision 1.2* detection and normal USB data communication, this field must remain 0. 0b - Disable 1b - Enable
1 DETECT_SEC	Secondary Detection Function Enable Enables the BC v1.2 secondary detection function for charger detect depending on other fields. If this field is 0, the BC secondary detection functions, when controlled by this register, are disabled. If this field is 1, the BC secondary detection functions, when controlled by this register, are enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	After a device determines that it is connected to a charging port during primary detection, you can use secondary detection to determine whether the remote port is a charging downstream port or a dedicated charging port. This field has an impact only when DCDSEL and EN_B are 0. The results of this detection are observed in USB1_CHRG_DET_STAT[SECDET_DCP]. 0b - Disable 1b - Enable
0 —	The value of this reserved bit must remain 0b to avoid interference with USB battery charging detection.

62.7.14 Charger Detect Status (USB1_CHRG_DET_STAT)

Offset

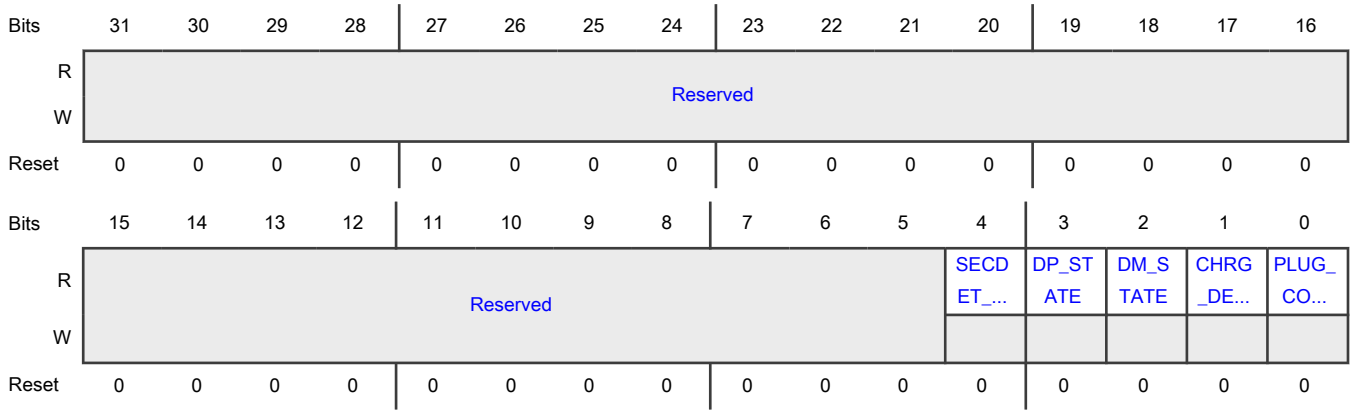
This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
USB1_CHRG_DET_STAT	F0h	Charger Detect Status
USB1_CHRG_DET_STAT_SET	F4h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_CHRG_DET_STAT is 1
USB1_CHRG_DET_STAT_CLR	F8h	Writing 1 to a bit in this register ensures that the corresponding bit in USB1_CHRG_DET_STAT is 0
USB1_CHRG_DET_STAT_TOG	FCh	Writing 1 to a bit in this register inverts the value of the corresponding bit in USB1_CHRG_DET_STAT

Function

Allows observation of status for USB charger detection functions.

Diagram



Fields

Field	Function
31-5 —	Reserved
4 SECDET_DCP	Battery Charging Secondary Detection Phase Output Indicates the type of charging port (charging downstream port (CDP) or downstream charging port (DCP)) detected during the secondary detection phase, according to *USB Battery Charging Specification, Revision 1.2*, using either Charger Detect (USB1_CHRG_DETECT) or the USBHSDCD module. 0b - CDP detected 1b - DCP detected
3 DP_STATE	DP Voltage Indicates the single-ended receiver output from charger detection circuits for the USB_DP pin. 0b - USB_DP pin voltage is ≤ 0.8 V 1b - USB_DP pin voltage is ≥ 2.0 V
2 DM_STATE	DM Voltage Indicates the single-ended receiver output from charger detection circuits for the USB_DM pin. 0b - USB_DM pin voltage is ≤ 0.8 V 1b - USB_DM pin voltage is ≥ 2.0 V
1 CHRG_DETECTED	Battery Charging Primary Detection Phase Output Indicates whether a standard downstream port (SDP) or a charging port is detected during the USB battery charging primary detection phase using either Charger Detect (USB1_CHRG_DETECT) or the USBHSDCD module. If a charging port is detected, this field reverts to 0 when primary detection ends and secondary detection starts. 0b - SDP detected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Charging port detected
0 PLUG_CONTACT	Battery Charging Data Contact Detection Phase Output Indicates whether an attached USB cable is detected between the remote host and the local device during the data contact detection phase, according to *USB Battery Charging Specification, Revision 1.2*, using either Charger Detect (USB1_CHRG_DETECT) or the USBHSDCD module. 0b - Not detected 1b - Detected

62.7.15 Analog Control (ANACTRL)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
ANACTRL	100h	Analog Control
ANACTRL_SET	104h	Writing 1 to a bit in this register ensures that the corresponding bit in ANACTRL is 1
ANACTRL_CLR	108h	Writing 1 to a bit in this register ensures that the corresponding bit in ANACTRL is 0
ANACTRL_TOG	10Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in ANACTRL

Function

Includes fields for:

- Analog control functions.
- Enabling the host pulldowns on the USB_DP and USB_DM pins when not in Host mode.
- Allowing control of an LVI detector.
- Configuring the PFD clock output. The PFD input clock comes from the 480 MHz USB PLL, and you use the PFD output clock (pfd_clk) to generate the USB1PFDCLK for use outside USBPHY.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	Reserved			Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		Reserved		Reserved	DEV_PUL...	Reserved	Reserved		Reserved			PFD_CLK_SEL		LVI_EN	Reserved
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-28 —	Reserved
27 —	The value of this reserved bit must remain 0b.
26 —	The value of this reserved bit must remain 0b.
25 —	The value of this reserved bit must remain 1b.
24 —	The value of this reserved bit must remain 0b.
23 —	The value of this reserved bit must remain 0b.
22 —	The value of this reserved bit must remain 0b.
21 —	The value of this reserved bit must remain 0b.
20 —	The value of this reserved bit must remain 0b.

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 —	The value of this reserved bit must remain 0b.
18 —	The value of this reserved bit must remain 0b.
17 —	The value of this reserved bit must remain 0b.
16 —	The value of this reserved bit must remain 0b.
15-14 —	To avoid unexpected USB behavior, do not change the value of this field without a specific recommendation from NXP.
13 —	To avoid unexpected USB behavior, do not change the value of this field without a specific recommendation from NXP.
12-11 —	To avoid unexpected USB behavior, do not change the value of this field without a specific recommendation from NXP.
10 DEV_PULLDOWN	Device Pulldown Enable Enables the 15 kΩ pulldown resistors on both USB_DP and USB_DM pins. You can use this feature in Device mode, when the USB cable is disconnected, to keep the data pins at known values, avoiding unnecessary interrupts from the single-ended receivers. The field is 1 at chip reset to help prevent noise on the USB cable before the chip is initialized. You must write 0 to this field during normal USB data communication in Device mode, or during battery charger detection, using the USBHSDCD module. Writing 1 to this field unconditionally enables both of the pulldown resistors. The effect of this field does not depend on the Host mode signals from the controller or on the state of DEBUG0[ENHSTPULLDOWN]. 0b - Disable 1b - Enable
9 —	Reserved
8-7 —	The value of this reserved bit must remain 00b.
6-4 —	The value of this reserved bit must remain 000b.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 PFD_CLK_SEL	PFD Clock Selection Selects the frequency relationship between the local pfd_clock and USB1PFDCLK outputs exported from USBPHY. See PFD A (PFDA) for other configuration information for the PFD clock. If this field is 00b, USB1PFDCLK has the same frequency as the USB PLL reference clock. If this field is 01b, the USB1PFDCLK frequency is pfd_clk divided by 4. If this field is 10b, the USB1PFDCLK frequency is pfd_clk divided by 2. If this field is 11b, USB1PFDCLK is the same as the pfd_clk frequency. 00b - USB1PFDCLK = USB PLL reference clock 01b - USB1PFDCLK = pfd_clk ÷ 4 10b - USB1PFDCLK frequency = pfd_clk ÷ 2 11b - USB1PFDCLK = pfd_clk
1 LVI_EN	Internal Low Voltage Detector Enable Enables an internal low-voltage detector for USBPHY's 3.3 V and 1.8 V power inputs. The low-voltage detector is used for status monitoring and isolation logic. See STATUS[OK_STATUS_3V] for more information. Writing 1 to this field enables this low-voltage detector. **NOTE** You must write 1 to this field for the CLN40ULP PHY. 0b - Disable 1b - Enable
0 —	The value of this reserved bit must remain 0b.

62.7.16 Trim (TRIM_OVERRIDE_EN)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
TRIM_OVERRIDE_EN	130h	Trim
TRIM_OVERRIDE_EN_SET	134h	Writing 1 to a bit in this register ensures that the corresponding bit in TRIM_OVERRIDE_EN is 1
TRIM_OVERRIDE_EN_CLR	138h	Writing 1 to a bit in this register ensures that the corresponding bit in TRIM_OVERRIDE_EN is 0
TRIM_OVERRIDE_EN_TOG	13Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in TRIM_OVERRIDE_EN

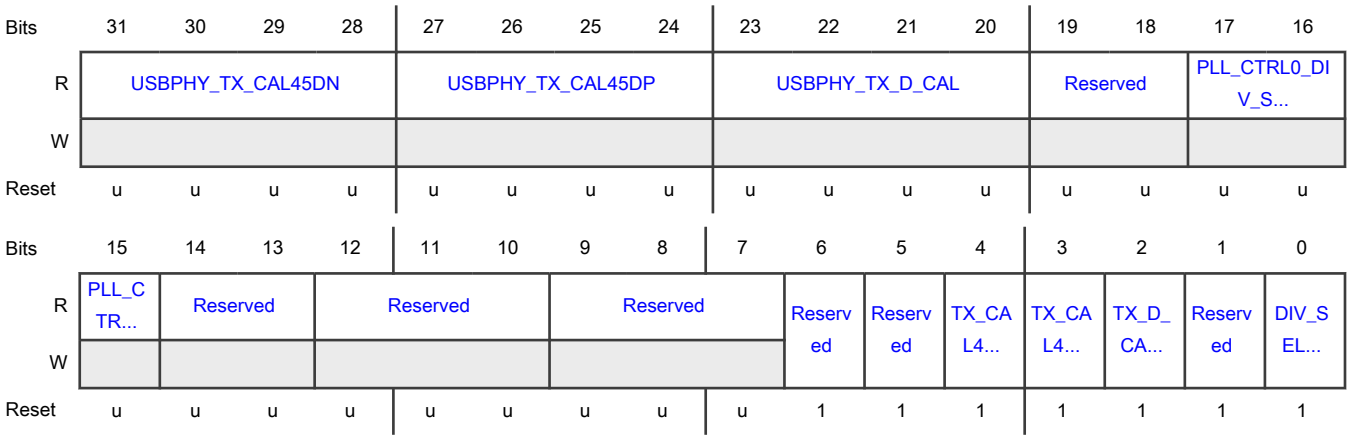
Function

Allows observation of the default nonvolatile settings of USBPHY parameter controls for USBPHY_TX_CAL45DN, USBPHY_TX_CAL45DP, USBPHY_TX_D_CAL, and PLL_CTRL0_DIV_SEL.

Additional fields also determine whether those values are overridden by using the settings defined in USBPHY's TX Control (TX) and PLL SIC (PLL_SIC). The default behavior is to use the values in TX Control (TX) and PLL SIC (PLL_SIC) instead of this register.

The method used to load the read-only values into this register during reset and boot is chip-specific. The values observed in the read-only fields after boot may vary from the reset values shown.

Diagram



Fields

Field	Function
31-28 USBPHY_TX_CAL45DN	DM Series Termination Resistance Trim Bits from Outside USBPHY Decodes to trim the nominal 45 Ω series termination resistance to the USB_DM output pin, with values loaded using signals outside USBPHY. When TX_CAL45DM_OVERRIDE is 1, the values in this field are used to trim the termination resistance. When TX_CAL45DM_OVERRIDE is 0, the resistance trim values in USBPHY_TX_CAL45DN are used instead. The methods used to load the read-only values into this field during reset and boot are chip-specific. Maximum resistance is at value 0000b, minimum resistance is at value 1111b, and resistance is centered at value 0111b. For this USBPHY, each incrementally increasing trim value decreases the target resistance by about 3.5% compared to the next lower value. Trimming this resistance impacts both the overshoot and undershoot of the FS TX output and the amplitude of the HS TX output. The methods used to load the read-only values into this register during reset and boot are chip-specific.
27-24 USBPHY_TX_CAL45DP	DP Series Termination Resistance Trim Bits from Outside USBPHY Decodes to trim the nominal 45 Ω series termination resistance to the USB_DP output pin, with values loaded using signals outside USBPHY. When TX_CAL45DP_OVERRIDE is 1, the values in this field are used to trim the termination resistance. When TX_CAL45DP_OVERRIDE is 0, the resistance trim values in USBPHY_TX_CAL45DP are used instead. The methods used to load the read-only values into this field during reset and boot are chip-specific.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Maximum resistance is at value 0000b, minimum resistance is at value 1111b, and resistance is centered at value 0111b. For this USBPHY, each incrementally increasing trim value decreases the target resistance by about 3.5% compared to the next lower value. Trimming this resistance impacts both the overshoot and undershoot of the FS TX output and the amplitude of the HS TX output.
23-20 USBPHY_TX_D_CAL	HS TX Output Current Trim Bits from Outside USBPHY Decodes to trim the nominal 17.78 mA current source for the HS TX drivers on USB_DP and USB_DM, with values loaded using signals from outside USBPHY. This output current is directly proportional to the amplitude of the HS TX eye diagram. When TX_D_CAL_OVERRIDE is 1, the values in this field are used to trim the output current. When TX_D_CAL_OVERRIDE is 0, the current trim values in TX[D_CAL] are used instead. The methods used to load the read-only values into this field during reset and boot are chip-specific. 0000b - Maximum current, approximately 19% above nominal 0111b - Nominal 1111b - Minimum current, approximately 19% below nominal
19-18 —	The values observed for this field may vary.
17-15 PLL_CTRL0_DIV_SEL	PLL Divider Value Configuration Bits from Outside USBPHY Controls the USB PLL feedback loop divider and allows the use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator, with values loaded from outside USBPHY. USBPHY's PLL produces a 480 MHz output clock. When DIV_SEL_OVERRIDE is 1, the values in this field are used to control the PLL divider. When DIV_SEL_OVERRIDE is 0, the PLL divider values in PLL_SIC[PLL_DIV_SEL] are used instead. See the field value descriptions for PLL_SIC[PLL_DIV_SEL] to understand the available input frequency settings. The methods used to load the read-only values into this field during reset and boot are chip-specific. The input clock frequency used for the PLL reference clock input is the output of the pre-divider stage controlled by PLL_SIC[PLL_PREDIV] bit field. For most SOCs the pre-divider would not be enabled but this feature might be used if an SOC had a very high frequency but still stable system reference clock.
14-13 —	The values observed for this field may vary.
12-10 —	The values observed for this field may vary.
9-7 —	Reserved
6	The value of this reserved bit must remain 1b.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 —	The value of this reserved bit must remain 1b.
4 TX_CAL45DM_OVERRIDE	Override Enable for USB_DM Series Termination Trim Determines whether the values of USBPHY_TX_CAL45DN or TX[TXCAL45DN] are used for the DM output resistance adjustment. If this field is 0, use the values from Trim (TRIM_OVERRIDE_EN) for the bandgap voltage trim. If this field is 1, use the values of TX Control (TX) for DM resistance trim. 0b - TRIM_OVERRIDE_EN 1b - TX
3 TX_CAL45DP_OVERRIDE	Override Enable for USB_DP Series Termination Trim Determines whether the values of USBPHY_TX_CAL45DP or TX[TXCAL45DP] are used for the DP output resistance adjustment. If this field is 0, use the values from Trim (TRIM_OVERRIDE_EN) for DP resistance trim. If this field is 1, use the values of TX Control (TX) for DP resistance trim. 0b - TRIM_OVERRIDE_EN 1b - TX
2 TX_D_CAL_OVERRIDE	Override Enable for HS TX Output Current Trim Determines whether the values of USBPHY_TX_D_CAL or TX[D_CAL] are used for the HS TX current adjustment. If this field is 0, use the values from Trim (TRIM_OVERRIDE_EN) for DP resistance trim. If this field is 1, use the values of TX Control (TX) for DP resistance trim. 0b - TRIM_OVERRIDE_EN 1b - TX
1 —	The value of this reserved bit must remain 1b.
0 DIV_SEL_OVERRIDE	Override Enable for PLL Divider Value Determines whether the values of PLL_CTRL0_DIV_SEL or PLL_SIC[PLL_DIV_SEL] are used for the USB PLL feedback loop divider. If this field is 0, use the values from Trim (TRIM_OVERRIDE_EN) for the PLL divider value. If this field is 1, use the values of PLL SIC (PLL_SIC) for the PLL divider value. 0b - TRIM_OVERRIDE_EN 1b - PLL_SIC

62.7.17 PFD A (PFDA)

Offset

This type of register has supplemental _SET, _CLR, and _TOG registers at adjacent offsets.

Register	Offset	Description
PFDA	140h	PFD A
PFDA_SET	144h	Writing 1 to a bit in this register ensures that the corresponding bit in PFDA is 1
PFDA_CLR	148h	Writing 1 to a bit in this register ensures that the corresponding bit in PFDA is 0
PFDA_TOG	14Ch	Writing 1 to a bit in this register inverts the value of the corresponding bit in PFDA

Function

Includes fields for the PFD output clock.

The PFD input clock comes from the 480 MHz USB PLL, and uses the PFD output clock (pfd_clk) to generate USB1PFDCLK for use outside USBPHY.

There is 1 set of PFD controls and PFD output clock on this chip.

ANACTRL[PFD_CLK_SEL] affects the PFD output clock behavior.

To configure the PFD clock:

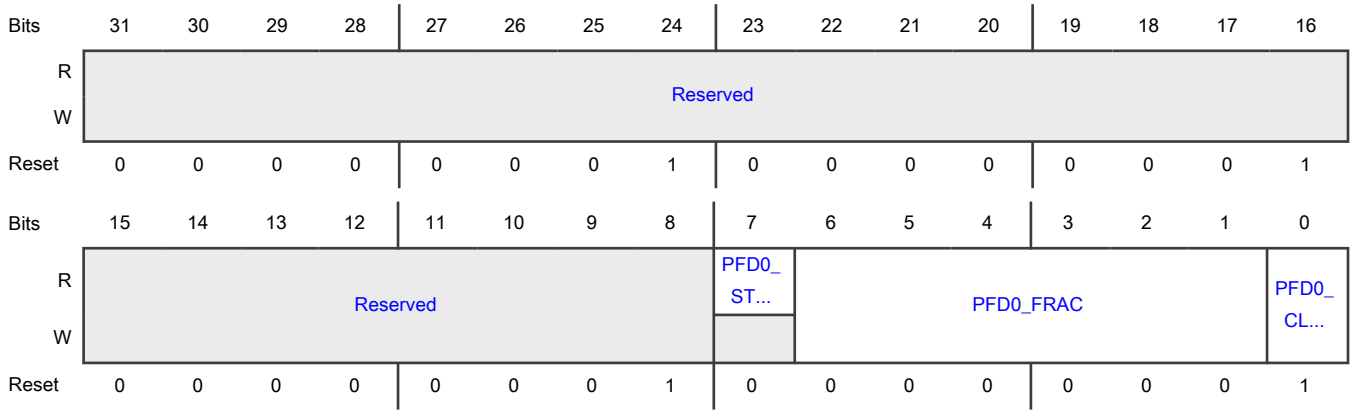
1. Wait until PLL_SIC[PLL_LOCK] becomes 1.
2. Configure ANACTRL[PFD_CLK_SEL] to select the PFD clock source.
3. Configure PFD*x*_FRAC to select the PFD clock frequency.
4. Write 0 to PFD*x*_CLKGATE to ungate the PFD clock.
5. Wait until PFD*x*_STABLE becomes 1.

Now you can use the PFD clock for system functions.

NOTE

If you want to change the value of PFD*x*_FRAC, gate the PFD clock (write 1 to PFD*x*_CLKGATE) and reset the PLL, that is, write 0 to PLL_SIC[PLL_POWER] and then write 1 to it again. After the reset, follow the aforementioned steps to reconfigure PFD*x*_FRAC.

Diagram



Fields

Field	Function
31-8 —	Reserved
7 PFD0_STABLE	PFD0 Stable Signal Allows PFD output clock frequency changes without forcing the relock of the USBPHY PLL. This field has an initial value of 0 after reset and transitions to a value of 1, indicating that the PFD clocks have stabilized following a logic path-settling interval after PFD0_FRAC is updated to change the fractional divider value. The value of this field continues to invert each time PFD0_FRAC is changed without resetting the PLL. This is a debug and diagnostic field that must not be used in the USB software stack because the PFD clock output becomes stable very quickly after the fractional divider value is changed. 0b - Not stable 1b - Stable
6-1 PFD0_FRAC	PFD0 Fractional Divider Selects the pfd_clk output frequency. Changing this field causes pfd_update to be toggled. The pfd_clk output frequency is: 480 MHz × 18 ÷ *n*, where *n* = 18–35 Valid values of PFD0_FRAC range from 0010_0110_1010b to 0010_0111_1011b; other values may cause undefined results.
0 PFD0_CLKGATE	PFD0 Clock Gate Controls clock gating (disabling) for the PFD pfd_clk output for power savings when the PFD is not used. Write 1 to this field to gate the output of the PFD. If the PFD reference clock is not stable, the USB PLL is not locked, or the USB PLL is not enabled, this gating is also enabled. This field must be reset to 0 to enable the specified pfd_clk output. 0b - Enable 1b - Disable

62.8 Register macro usage

A read-modify-write (RMW) operation involves updating one field without disturbing the contents of the remaining fields in the register. In this operation, the CPU reads the register, modifies the target field, and then writes the results back to the register. Because of the initial register read, this is an expensive operation in terms of CPU cycles.

To address this issue, you can implement some USBPHY registers as a group, including registers that can be used to either set, clear, or toggle (SCT) individual fields of the primary register. When writing to an SCT register, all fields that are 1 perform the associated operation on the primary register, while all fields that are 0 are not affected. The SCT registers always read back 0 and must have the write-only access type. If the primary register is read-only, you cannot implement SCT registers.

With this architecture, you can update one or more fields using only register writes. First, a write to the associated clear register clears all bits of the target fields, and then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have a potential drawback—whenever a field is modified, USBPHY sees a value of 0 before you write the final value. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when performing a CS operation.

Also, you do not require a CS operation for fields that are 1-bit wide. When the CS operation works in this case, it is more efficient to simply set or clear the target bit (that is, one write instead of two). A simple set or clear operation is also atomic, whereas a CS operation is not.

All macros for SCT are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of RMW operations. When an atomic operation is required, you must remember this, because if an interrupt occurs during a critical update sequence, unexpected behavior may take place.

62.9 References

The following publications are referenced in this document. For updates, see www.usb.org.

- *Universal Serial Bus Specification, Revision 2.0*, 2000, with amendments including those listed below
- *Errata for "USB Revision 2.0 April 27, 2000" as of 12/7/2000*
- *Errata for "USB Revision 2.0 April 27, 2000" as of May 28, 2002*
- *Pull-up / Pull-down Resistors* (USB Engineering Change Notice)
- *Suspend Current Limit Changes* (USB Engineering Change Notice)
- *Device Capacitance* (USB Engineering Change Notice)
- *USB 2.0 Connect Timing Update* (USB Engineering Change Notice as of April 4, 2013)
- *USB 2.0 VBUS Max Limit* (USB Engineering Change Notice)
- *On-The-Go and Embedded Host Supplement to the USB Revision 2.0 Specification*, Revision 2.0 version 1.1a, July 27, 2012
- *Maximum VBUS Voltage* (USB OTGEH Engineering Change Notice)
- *Universal Serial Bus Micro-USB Cables and Connectors Specification*, Revision 1.01, 2007

Chapter 63
USB Device Charger Detection Module (USBDCD)

63.1 Chip-specific USBDCD information

Table 493. Reference links to related information

Topic	Related module	Reference
Full description	USBDCD	USBDCD
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
Interrupts	NVIC	Interrupts **NOTE** For USBFSDCD, the interrupt is same as USBFS, and for USBHSDCD, it is same as USB HS PHY.

63.1.1 Module instances

This chip includes two USBDCD blocks.

- USBDCD is used for control of device charger detect for USB FS/LS DCD function on USB0 port.
- USBHSDCD is used for control of device charger detect for USB HS/FS/LS DCD function on USB1 port.

63.1.2 Reset control

- USBDCD: Reset control is same as USBFS.
- USBHSDCD: Reset control is same as USB HS PHY.

See Peripheral Reset Control 1 (PRESETCTRL1) and Peripheral Reset Control 2 (PRESETCTRL2).

63.1.3 AHB clock control

- USBDCD: Clock control is same as USBFS.
- USBHSDCD: Clock control is same as USB HS PHY.

See AHB Clock Control 1 (AHBCLKCTRL1) and AHB Clock Control 2 (AHBCLKCTRL2).

63.2 Overview

USBDCD works with the USB transceiver to detect whether the USB device is attached to a Charging Port, either a Dedicated Charging Port (DCP) or a Charging Downstream Port (CDP). System software coordinates the detection activities of the module and controls an off-chip integrated circuit that performs the battery charging.

The use of the term "USB transceiver" in this module documentation applies to the USB physical layer instance on the chip, whether it is an Full Speed (FS)/Low Speed (LS) only transceiver or an High Speed (HS)/FS/LS capable PHY.

63.2.1 Block diagram

The following figure is a high-level block diagram of the module.

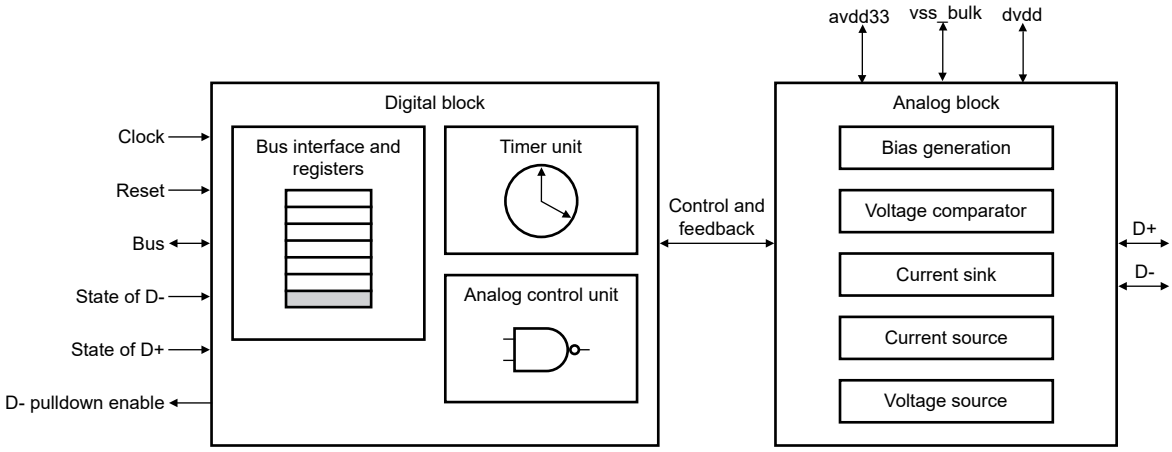


Figure 343. Block diagram

USBDCD consists of two main blocks:

- A Digital block provides the programming interface (memory-mapped registers) and includes the Timer unit and the Analog control unit.
- An Analog block provides the circuitry for the physical detection of the charger that includes the Bias generation, Voltage source, Current source, Current sink, and Voltage comparator circuitry. This block also contains necessary muxes between the source/sink circuits and the D- and D+ pins.

63.2.2 Features

- Compliant with the latest industry-standard specifications: *USB Battery Charging Specification, Revisions 1.1 and 1.2*
- Supports programmable timing parameters:
  - By default set to values required by the industry standards to allow easy configuration. You only need to set the clock frequency before enabling the module.
  - Updates of the standards are programmable.

63.3 Functional description

Several hardware components are involved in the sequence of detecting the presence of the charging port and type of the charging port. System software coordinates this activity. This collection of interacting hardware and software is called USB battery charging subsystem. The following figure shows USBDCD as a component of the subsystem.

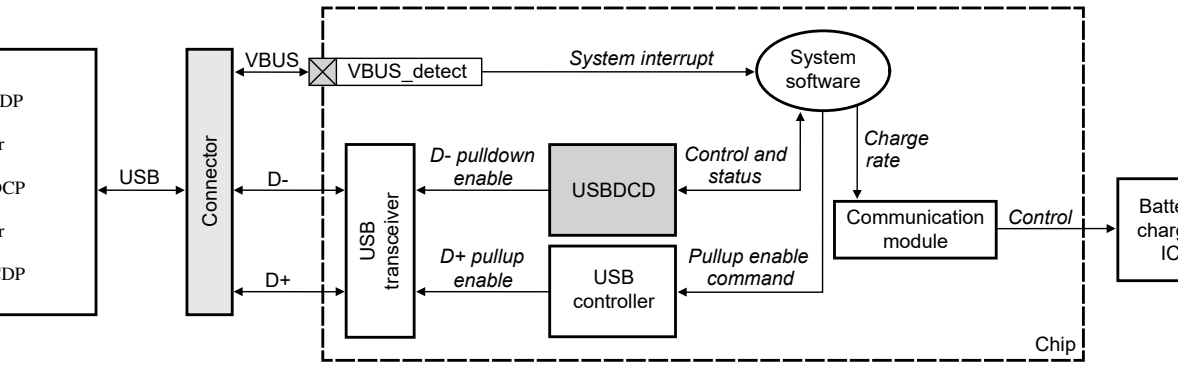


Figure 344. USB battery charging subsystem

The following table describes the components.

Table 494. USB battery charger subsystem components

<table>
<tr><th>Component</th><th>Description</th></tr>
<tr><td>Battery Charger IC</td><td>The external battery charger IC regulates the charge rate to the rechargeable battery. System software is responsible for communicating the appropriate charge rates.
<table>
<tr><th>Charger</th><th>Maximum current drawn</th></tr>
<tr><td>Standard Downstream Port (SDP)</td><td>up to 500 mA (I_{CFG_MAX})[1]</td></tr>
<tr><td>CDP</td><td>up to 1500 mA (I_{DEV_CHG})</td></tr>
<tr><td>DCP</td><td>up to 1500 mA (I_{DEV_CHG})</td></tr>
</table>
1. If a USB SDP host has suspended the USB device, the current drawn by the device has to follow the rules of the Dead Battery Provision in the USB Battery Charging Specification, Rev. 1.2.</td></tr>
<tr><td>Comm Module</td><td>A communications module on the device can be used to control the charge rate of the battery charger IC.</td></tr>
<tr><td>System software</td><td>Coordinates the detection activities of the subsystem.</td></tr>
<tr><td>USB Controller</td><td>The D+ pullup enable control signal plays a role during the charger type detection phase. System software must issue a command to the USB controller to assert this signal. After this pullup is enabled, the device is considered to be connected to the USB bus. The host then attempts to enumerate it.
NOTE
In general, the USB controller is used only for USB device applications when using most features of USBDCD. The exception is for USB host applications which need to advertise CDP capability. See Host Mode CDP advertisement signaling. Otherwise, USBDCD must be disabled in host mode.</td></tr>
<tr><td>USB Transceiver</td><td>The USB transceiver contains the pullup resistor for the USB D+ signal and the pulldown resistors for the USB D+ and D– signals. The D+ pullup and the D– pulldown are both used during the charger detection sequence in BC1.1, but it is not used during charger detection in BC1.2. The USB transceiver also outputs the digital state of the D+ and D– signals from the USB bus.
The pullup and pulldown enable signals are controlled by other modules during the charger detection sequence in BC1.1: The D+ pullup enable is output from the USB controller and is under software control. USBDCD controls the D– pulldown enable.</td></tr>
<tr><td>USBDCD</td><td>Detects whether the device has been plugged into either an SDP, a CDP, or a DCP.</td></tr>
<tr><td>VBUS_detect</td><td>This interrupt pin connected to the USB VBUS signal detects when the device has been plugged into or unplugged from the USB bus. If the system requires waking up from a low-power mode on being plugged into the USB port, this interrupt should also be a low-power wakeup source. If this pin multiplexes other functions, such as GPIO, the pin can be configured as an interrupt so that the USB plug or unplug event can be detected.</td></tr>
</table>

63.3.1 Modes of operation

The operating modes of the USBDCD module are shown in the following table.

Table 495. Module modes and their conditions

Module mode	Description	Conditions when used
Enabled for Charger Detection	The module performs the charger detection sequence.	System software must enable the module only if all of the following conditions are true: • The system is using a rechargeable battery or is otherwise capable of being powered up by an SDP or a Charging Port. • The device is being used in a USB device application. • The device is attached to the USB cable.
Enabled for Signal Overrides	Module bias circuits are enabled for special signaling on DP/DM pins.	See SIGNAL_OVERRIDE[PS].
Disabled	The module is not active and is held in a low-power state.	System software must disable the module when either of the following conditions is true: • The charger detect sequence is complete. • The conditions for being enabled are not met.
Powered Off	The digital supply voltage dvdd is removed.	Low system performance requirements allow putting the device into a very low-power stop mode.

Operating mode transitions are shown in the following table.

Table 496. Entering and exiting module modes

Module mode	Entering	Exiting	Mode after exiting
Enabled	Set CONTROL[START].	Set CONTROL[SR]. [1]	Disabled
Disabled	Take *either* of the following actions: • Set CONTROL[SR].[1] • Reset the module. By default, the module is disabled.	Set CONTROL[START].	Enabled
Powered Off	Perform the following actions: 1. Put the device into very low-power stop mode. 2. Adjust the supply voltages.	Perform the following actions: 1. Restore the supply voltages. 2. Take the device out of very low-power stop mode.	Disabled

1. The effect of setting the CONTROL[SR] field is immediate; that is, the module is disabled even if the sequence has not completed.

63.3.2 The charger detection sequence

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the *USB Battery Charging Specification, Rev. 1.2*.

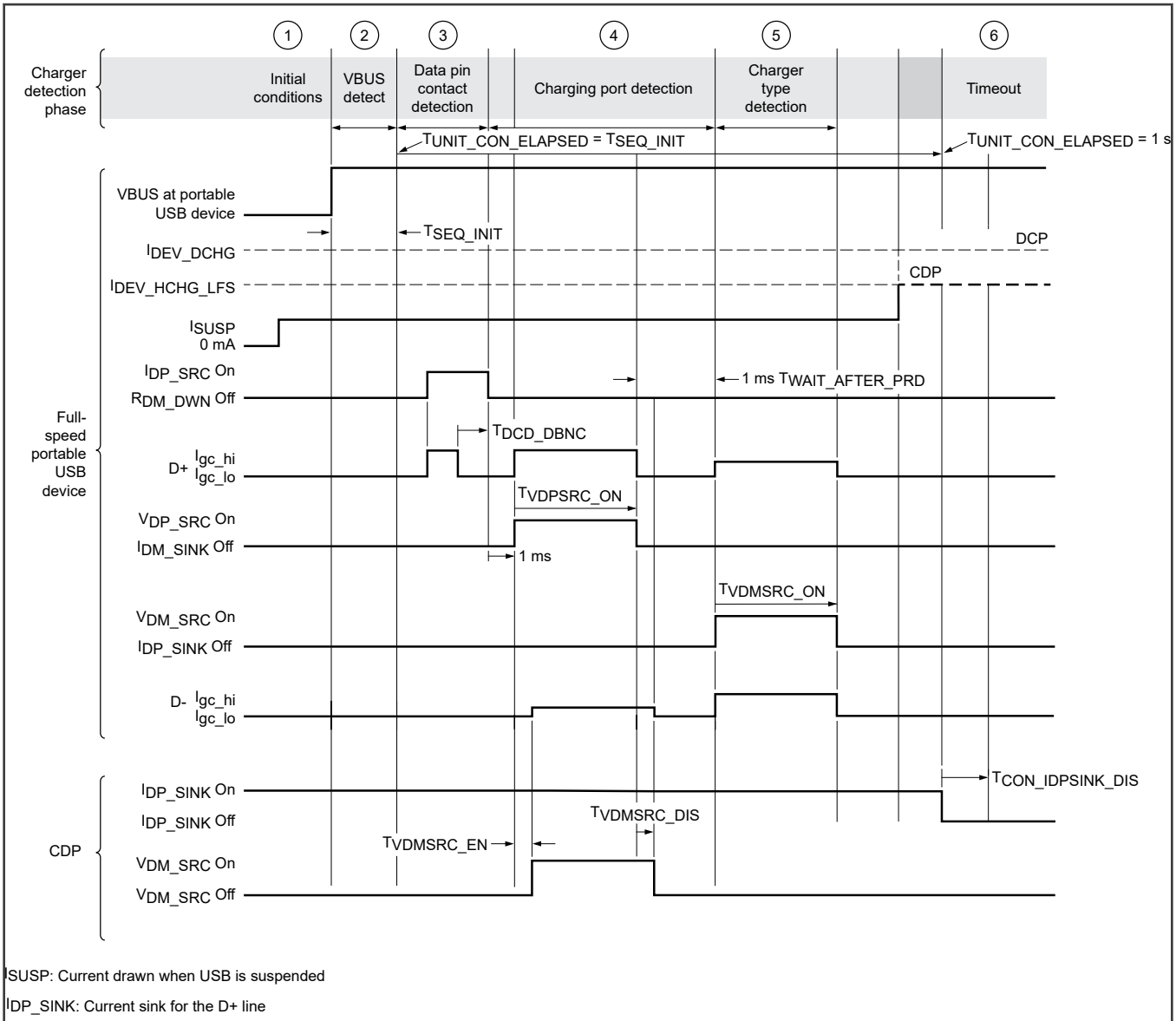


Figure 345. FS charger detection timing for BC1.2

Timing parameter values used in this module for BC1.2 are listed in the following table.

Table 497. Timing parameters for the charger detection sequence for BC1.2

Parameter	USB Battery Charging Spec	Module default	Module programmable range
T_{DCD_DBNC} [1]	10 ms min (no max)	10 ms	0– 1023 ms
T_{VDPSRC_ON} [1]	40 ms min (no max)	40 ms	0 –1023 ms
$T_{WAIT_AFTER_PRD}$	N/A	1 ms	0– 1023 ms
T_{VDMSRC_ON}	40 ms	40 ms	0 –1023 ms

Table continues on the next page...

Table 497. Timing parameters for the charger detection sequence for BC1.2 (continued)

Parameter	USB Battery Charging Spec	Module default	Module programmable range
T_{SEQ_INIT}	N/A	16 ms	0 –1023 ms
T_{UNIT_CON} [1]	1 s	N/A	N/A
T_{VDMSRC_EN} [1]	1– 20 ms	From the USB host	N/A
T_{VDMSRC_DIS} [1]	0 – 20 ms	From the USB host	N/A
$T_{CON_IDPSINK_DIS}$ [1]	0 – 10 ms	From the USB host	N/A

1. This parameter is defined by the *USB Battery Charging Specification, Rev. 1.2*.

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the *USB Battery Charging Specification, Rev. 1.1*.

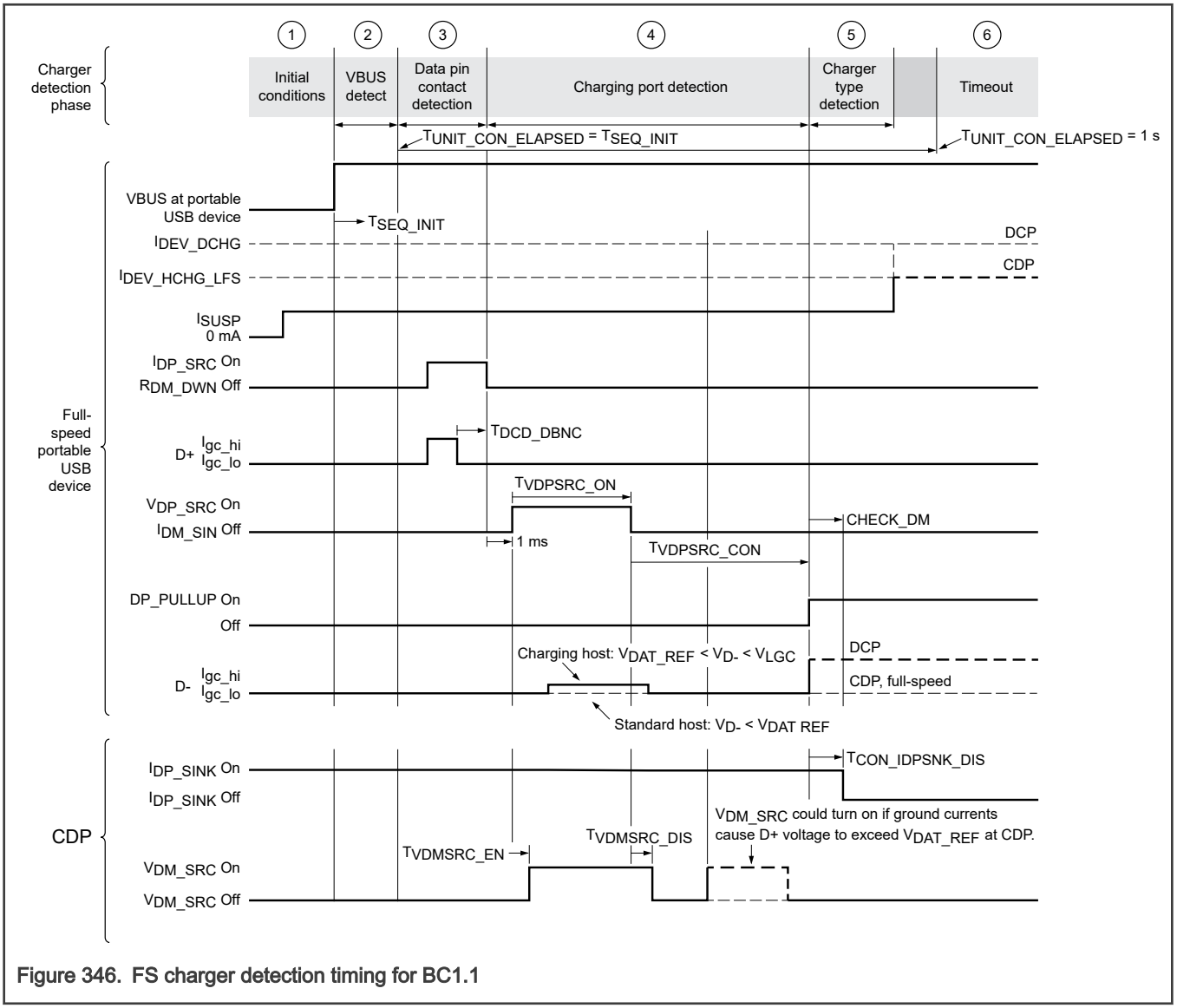


Figure 346. FS charger detection timing for BC1.1

Timing parameter values used in this module for BC1.1 are listed in the following table.

Table 498. Timing parameters for the charger detection sequence for BC1.1

Parameter	USB Battery Charging Spec	Module default	Module programmable range
T_{DCD_DBNC} [1]	10 ms min (no max)	10 ms	0– 1023 ms
T_{VDPSRC_ON} [1]	40 ms min (no max)	40 ms	0 –1023 ms
T_{VDPSRC_CON} [1]	40 ms min (no max)	40 ms	0 –1023 ms
CHECK_DM	N/A	1 ms	0– 15 ms
T_{SEQ_INIT}	N/A	16 ms	0 –1023 ms
T_{UNIT_CON} [1]	1 s	N/A	N/A
T_{VDMSRC_EN} [1]	1– 20 ms	From the USB host	N/A
T_{VDMSRC_DIS} [1]	0 –20 ms	From the USB host	N/A
$T_{CON_IDPSINK_DIS}$ [1]	0– 20 ms	From the USB host	N/A

1. This parameter is defined by the *USB Battery Charging Specification, Rev. 1.1*.

The following table provides an overview description of the charger detection sequence shown in the preceding figure.

Table 499. Overview of the charger detection sequence

Phase		Overview description	Full description
1	Initial conditions	Initial system conditions that need to be met before the detection sequence is initiated.	Initial system conditions
2	VBUS detection	System software detects contact of the VBUS signal with the system interrupt pin VBUS_detect.	VBUS contact detection
3	Data pin contact detection	The USBDCD module detects that the USB data pins D+ and D– have made contact with the USB port.	Data pin contact detection
4	Charging port detection	The USBDCD module detects if the port is an SDP or either type of charging port, that is CDP or DCP.	Charging port detection
5	Charger type detection	The USBDCD module detects the type of charging port, if applicable.	Charger type detection
6	Sequence timeout	The USBDCD module did not finish the detection sequence within the timeout interval. The sequence will continue until halted by software.	Charger detection sequence timeout

63.3.2.1 Initial system conditions

The USBDCD module is intended for use with USB device applications using a rechargeable battery. The module does not have support for interfacing with ACA or ACA-Dock equipment as defined in the *USB Battery Charging Specification,*

Revision 1.2. It cannot be used with USB applications that are embedded host or OTG, except as indicated in Host Mode CDP advertisement signaling.

In addition, before the USBDCD module's charger detection sequence can be initiated, the system must be:

- Powered-up and in run mode. The USBDCD instantiation of this module for the High-Speed port does not directly depend on the VBUS voltage and can operate as long as the avdd33 supply is in a valid range.
- Recently plugged into a USB port.
- Drawing not more than 2.5 mA total system current from the USB bus, except as allowed by the *USB 2.0 Connect Timing Update ECN*.

Examples of allowable precursors to this set of initial conditions include:

- A powered-down device is subsequently powered-up upon being plugged into the USB bus.
- A device in a low-power mode subsequently enters run mode upon being plugged into the USB bus.

63.3.2.2 VBUS contact detection

Once the device is plugged into a USB port, the VBUS_detect system interrupt is triggered. System software must do the following to initialize the module and start the charger detection sequence:

1. Restore power if the module is powered-off.
2. Set CONTROL[SR] to initiate a software reset.
3. Configure the USBDCD module by programming the CLOCK register and the timing parameters as needed.
4. Set CONTROL[IE] to enable interrupts, or clear CONTROL[IE] if the software polling method is used.
5. Program CONTROL[BC12] based on which revision of the USB Battery Charging Specification is needed.
6. Set CONTROL[START] to start the charger detection sequence.

63.3.2.3 Data pin contact detection

The module must ensure that the data pins have made contact because the detection sequence depends upon the state of the USB D+ and D- signals. USB plugs and receptacles are designed such that when the plug is inserted into the receptacle, the power pins make contact before the data pins make contact. See the following figure.

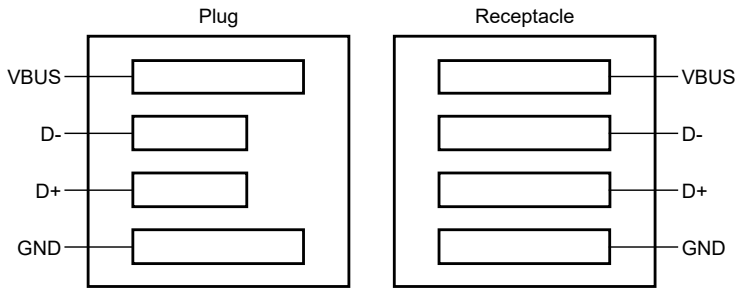


Figure 347. Relative pin positions in USB plugs and receptacles

As a result, when a portable USB device is attached to an upstream port, the portable USB device detects VBUS before the data pins have made contact. The time between power pins and data pins making contact depends on how fast the plug is inserted into the receptacle. Delays of several hundred milliseconds are possible.

63.3.2.3.1 Debouncing the data pin contact

When system software has initiated the charger detection sequence, as described in Initial system conditions, the USBDCD module turns on the I_{DP_SRC} current source and enables the R_{DM_DWN} pulldown resistor. If the data pins have not made contact, the D+ line remains high. After the data pins make contact, the D+ line goes low and debouncing begins.

After the D+ line goes low, the module continuously samples the D+ line over the duration of the T_{DCD_DBNC} debounce time interval. By default, T_{DCD_DBNC} is 10 ms, but it can be programmed in the TIMER0[TDCD_DBNC] field. See the description of TIMER0 for details.

When it has remained low for the entire interval, the debouncing is complete. However, if the D+ line returns high during the debounce interval, the module waits until the D+ line goes low again to restart the debouncing. This cycle repeats until either of the following happens:

- The data pin contact has been successfully debounced (see Success in detecting data pin contact (phase completion)).
- A timeout occurs (see Charger detection sequence timeout).

63.3.2.3.2 Success in detecting data pin contact (phase completion)

After successfully debouncing the D+ state, the module does the following:

- Updates the STATUS register to reflect phase completion (See Table 505 for field values.)
- Directly proceeds to the next step in the sequence: detection of a charging port (See Charging port detection.)

63.3.2.4 Charging port detection

After it detects that the data pins have made contact, the module waits for a fixed delay of 1 ms, and then attempts to detect whether it is plugged into a charging port. The module connects the following analog units to the USB D+ or D– lines during this phase:

- The voltage source V_{DP_SRC} connects to the D+ line
- The current sink I_{DM_SINK} connects to the D– line
- The voltage comparator connects to the USB D– line, comparing it to the voltage V_{DAT_REF}.

After a time of T_{VDPSRC_ON}, the module samples the D– line. The T_{VDPSRC_ON} parameter is programmable and defaults to 40 ms. After sampling the D– line, the module disconnects the voltage source, current sink, and comparator.

The next steps in the sequence depend on the voltage on the D– line as determined by the voltage comparator. See the following table.

Table 500. Sampling D– in the charging port detection phase

If the voltage on D- is...	Then...	See...
Below V_{DAT_REF}	The port is an SDP that does not support using charging currents above I_{CFG_MAX}.	SDP
Above V_{DAT_REF} but below V_{LGC}	The port is a charging port.	Charging port
Above V_{LGC}	This is an error condition.	Error in charging port detection

63.3.2.4.1 SDP

As part of the charger detection handshake with a standard USB host, the module does the following without waiting for the interval ($T_{WAIT_AFTER_PRD}$ or T_{VDPSRC_CON}) to elapse:

- Updates the STATUS register to reflect that an SDP is detected with SEQ_RES = 01b. See Table 505 for field values.
- Sets CONTROL[IF].

- Generates an interrupt if enabled in CONTROL[IE].

At this point, control is passed to system software via the interrupt. The rest of the sequence, which detects the type of charging port, is not applicable, so software must perform the following steps:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see Table 494.

63.3.2.4.2 Charging port

As part of the charger detection handshake with any type of USB host, the module waits until the interval ($T_{WAIT_AFTER_PRD}$ or T_{VDPSRC_CON}) has elapsed before it does the following:

For *USB Battery Charging Specification, Rev. 1.2*:

- Enables V_{DM_SRC}.
- Updates the STATUS register to reflect that a charging port is detected with SEQ_RES = 10b. See Table 505 for field values.
- At this point the detection sequence progresses to Charger type detection without needing software interaction.

For *USB Battery Charging Specification, Rev. 1.1*:

- Updates the STATUS register to reflect that a charging port is detected with SEQ_RES = 10b. See Table 505 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].
- At this point, control is passed to system software via the interrupt. Software must:
  1. Read the STATUS register.
  2. Set CONTROL[IACK] to acknowledge the interrupt.
  3. Issue a command to the USB controller to pullup the USB D+ line.
  4. Wait for the module to complete the final phase of the sequence. See Charger type detection.

63.3.2.4.3 Error in charging port detection

For this error condition, the module does the following:

- Updates the STATUS register to reflect the error with SEQ_RES = 00b. See Table 505 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

Note that in this case the module does not wait for the interval ($T_{WAIT_AFTER_PRD}$ or T_{VDPSRC_CON}) to elapse.

At this point, control has been passed to system software via the interrupt. The rest of the sequence (detecting the type of charging port) is not applicable, so software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.

63.3.2.5 Charger type detection

For *USB Battery Charging Specification, Rev. 1.2*:

After the USBDCD module enables the V_{DM_SRC}, the module starts the T_{VDMSRC_ON} timer counting down the time interval programmed into the TIMER2[T_{VDMSRC_ON}] field.

Once the T_{VDMSRC_ON} timer is elapsed, the module samples the USB D+ line to determine the type of charger. See the following table.

Table 501. Sampling D+ in the charger type detection phase (BC1.2)

If the voltage on D+ is...	Then...	See...
High (D+>V_{DAT_REF})	The port is a DCP. [1]	DCP
Low (D+<V_{DAT_REF})	The port is a CDP. [2]	CDP

1. In a DCP, the D+ and D– lines are shorted together through a small resistor.
2. In a CDP, the D+ and D– lines are not shorted.

For *USB Battery Charging Specification, Rev. 1.1*:

After software enables the D+ pullup resistor, the module is notified automatically (via internal signaling) to start the CHECK_DM timer counting down the time interval programmed in the TIMER2[CHECK_DM] field.

After the CHECK_DM time is elapsed, the module samples the USB D– line to determine the type of charger. See the following table.

Table 502. Sampling D– in the charger type detection phase (BC1.1)

If the voltage on D– is...	Then...	See...
High	The port is a DCP. [1]	DCP
Low	The port is a CDP. [2]	CDP

63.3.2.5.1 DCP

For a DCP, the module does the following:

- Updates the STATUS register to reflect that a dedicated charging port is detected with SEQ_RES = 11b. See Table 505 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE] field.

The *USB Battery Charging Specification, Rev. 1.2* indicates that additionally if the detection sequence determines an attachment to a DCP, then the USB device should signal on the D+ pin by either enabling the D+ pullup resistor or enabling V_{DP_SRC}.

At this point, control has been passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Disable the USB controller to prevent transitions on the USB D+ or D– lines from causing spurious interrupt or wakeup events to the system.
3. Set CONTROL[IACK] to acknowledge the interrupt.
4. Set CONTROL[SR] to issue a software reset to the module.
5. Disable the module.

6. Enable signaling on the D+ pin. This can be done by enabling the Transceiver D+ pullup resistor through configuration of the register fields in the USB controller instance for this USB port. For USBHSDCD instantiation of USBDCD, this signaling can alternatively be done by enabling V_{DP_SRC} through setting the SIGNAL_OVERRIDE[PS] field to a value of 010b.
7. Communicate the appropriate charge rate to the external battery charger IC; see Table 494.

63.3.2.5.2 CDP

For a CDP, the module does the following:

- Updates the STATUS register to reflect that a CDP is detected with SEQ_RES = 10b. See Table 505 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

At this point, control is passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see Table 494.

63.3.2.6 Charger detection sequence timeout

The maximum time allowed to connect according to the *USB Battery Charging Specification* is one second. If the Unit Connection Timer reaches the one second limit and the sequence is still running as indicated by the STATUS[ACTIVE] field, the module does the following:

- Updates the STATUS register to reflect that a timeout error has occurred. See Table 505 for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].
- The detection sequence continues until explicitly halted by software setting CONTROL[SR].
- The Unit Connection Timer continues counting. See the description of TIMER0.

At this point, control is passed to the system software via the interrupt, which has two options: ignore the interrupt and allow more time for the sequence to complete, or halt the sequence. To halt the sequence, software must:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.

This timeout function is also useful in case software does not realize that the USB device is unplugged from USB port during the charger detection sequence. If the interrupt occurs but the V_{BUS_DETECT} input is low, software can disable and reset the module.

System software might allow the sequence to run past the timeout interrupt under these conditions:

1. The USB Battery Charging Spec is amended to allow more time. In this case, software should poll TIMER0[$T_{UNITCON}$] periodically to track elapsed time after 1s; or
2. For debug purposes.

Note that the $T_{UNITCON}$ register field will stop incrementing when it reaches its maximum value so it will not roll over to zero and start counting up again.

63.3.2.7 Dead Battery Provision signaling

If the system needs to operate under the Dead Battery Provision of the *USB Battery Charging Specification, Revision 1.2*, the USBDCD module must be configured to signal V_{DP_SRC} on the USB_DP pin.

For information on Dead Battery Provision conditions and signaling, see the Dead or weak battery later in this chapter.

63.3.2.8 Host Mode CDP advertisement signaling

If the system is configured in Embedded Host mode, and the system hardware external to the SOC is capable of serving as a CDP, this capability can be signaled by the USBDCD module.

As indicated in the *USB Battery Charging Specification, Revision 1.2*, one way that the CDP can behave for this case is to signal V_{DM_SRC} on the USB_DM pin while there is not a remote device connected to the host, and then disable V_{DM_SRC} shortly after a connection event.

The V_{DM_SRC} signaling can be enabled through setting the SIGNAL_OVERRIDE[PS] field to 100b.

If this feature is used, the USB subsystem software must control this field upon detecting USB connect and disconnect events.

63.3.3 Clocking

The following table describes the clock sources for the USBDCD module. Refer to the chip's clocking chapter for clock setting, configuration, and gating information.

Note these clocks share same name with the USB PHY.

Table 503. USBDCD clocks

Clock name	Description
MODULE_CLK	Peripheral clock
MODULE_CLK_S	Peripheral access clock

63.3.4 Resets

There are two ways to reset various register contents in this module: hardware resets and a software reset.

63.3.4.1 Hardware resets

Hardware resets originate at the system or device level and propagate down to the individual module level. They include start up reset, low-voltage reset, and all other hardware reset sources.

Hardware resets cause the register contents to be restored to their default state as listed in the register descriptions.

63.3.4.2 Software reset

A software reset re-initializes the module's status information, but leaves configuration information unchanged. The software reset allows software to prepare the module without needing to reprogram the same configuration each time the USB device is plugged into a USB port.

Setting CONTROL[SR] initiates a software reset. The following table shows all register fields that are reset to their default values by a software reset.

Table 504. Software reset and register fields affected

Register	Fields affected	Fields not affected
CONTROL[1]	IF	IE, START
STATUS	All	None
CLOCK	None	All
TIMER*n*	TUNITCON	All other

1. CONTROL[SR] and CONTROL[IACK] are self-clearing.

A software reset also returns all internal logic, timers, and counters to their reset states. If the module is already active (STATUS[ACTIVE] = 1b), a software reset stops the sequence.

NOTE

Software must always initiate a software reset before starting the sequence to ensure the module is in a known state.

63.3.5 Interrupts

The USBDCD module has an interrupt to alert system software of certain events, which are listed in the following table. All events except the Phase Complete event for the Data Pin Detection phase can trigger an interrupt.

Table 505. Events triggering an interrupt by sequence phase

Sequence phase	Event	Event description	STATUS fields [1]	Phase description
Data Pin Detection	Phase Complete	The module has detected data pin contact. *No interrupt occurs*: CONTROL[IF] = 0b.	ERR = 0b SEQ_STAT = 01b SEQ_RES = 00b TO = 0b	VBUS contact detection
Charging Port Detection	Phase Complete	The module has completed the process of identifying if the USB port is a charging port or not. **NOTE:** An interrupt is triggered at completion of this phase for BC 1.1 operation whether an SDP or a Charging port is detected. For BC 1.2 operation, if an SDP is detected an interrupt is triggered but if a Charging port is detected the sequence goes directly to the Charger Type Detection phase.	ERR = 0b SEQ_STAT = 10b SEQ_RES = 01b or 10b TO = 0b	Charging port detection
	Error	The module cannot identify the type of port because the D– line is above the USB's VLGC threshold.	ERR = 1b SEQ_STAT = 10b SEQ_RES = 00b TO = 0b	Error in charging port detection

Table continues on the next page...

Table 505. Events triggering an interrupt by sequence phase (continued)

Sequence phase	Event	Event description	STATUS fields [1]	Phase description
Charger Type Detection	Phase Complete	The module has completed the process of identifying the charger type detection. **NOTE:** The ERR flag always reads zero because no known error conditions are checked during this phase.	ERR = 0b SEQ_STAT = 11b SEQ_RES = 11b or 10b TO = 0b	Charger type detection
Sequence Timeout	Error	The timeout interval from the time the USB device attaches to a USB port until it connects has elapsed.	ERR = 1b SEQ_STAT = last value [2] SEQ_RES = last value [2] TO = 1b	Charger detection sequence timeout

1. See the description of the Status register for register information.
2. The SEQ_STAT and SEQ_RES fields retain the values held at the time of the timeout error.

63.3.5.1 Interrupt handling

Software reads which event caused the interrupt from the STATUS register during the interrupt service routine.

An interrupt is generated only if CONTROL[IE] is set. The CONTROL[IF] field is always set under interrupt conditions, even if CONTROL[IE] is cleared. In this case, software can poll CONTROL[IF] to determine if an interrupt condition is pending.

Writes to CONTROL[IF] are ignored. To reset CONTROL[IF], set CONTROL[IACK] to acknowledge the interrupt. Writing to CONTROL[IACK] when CONTROL[IF] is cleared has no effect.

63.4 External signals

This section describes the module signals. The following table shows a summary of module signals that interface with the pins of the device.

Table 506. Signal descriptions

Signal	Description	I/O
USB_DM	USB D– analog data signal. The analog block interfaces directly to the D– signal on the USB bus.	I/O
USB_DP	USB D+ analog data signal. The analog block interfaces directly to the D+ signal on the USB bus.	I/O
avdd33 [1]	3.3 V regulated analog supply	POWER
vss_bulk	Digital/Analog ground	POWER
dvdd	Core voltage digital supply	POWER

1. Voltage must be 3.3 V ± 10% for full functionality of the module. That is, the charger detection function does not work when this voltage is below 3.0 V, and the CONTROL[START] field should not be set.

NOTE

The transceiver module also interfaces to the USB_DM and USB_DP signals. Both modules and the USB host/hub use these signals as bidirectional, tristate signals.

Information about the signal integrity aspects of the lines including shielding, isolated return paths, input or output impedance, packaging, suggested external components, ESD, and other protections can be found in the USB 2.0 Specification and in Application information.

63.5 Initialization

This module is designed for minimal configuration while retaining significant programmability. To initialize USBDCD:

1. Initialize the CLOCK register to the actual system clock frequency, unless the default value already matches the system requirements.
2. Reset ANACTRL[DEV_PULLDOWN] to 0b when this module is in use for the proper operation of the USBHSDCD instantiation of USBDCD.

   For more information, see the "USB PHY Analog Control Register (ANACTRL)" section of the "Universal Serial Bus 2.0 Integrated PHY (USB-PHY)" chapter.
3. Do not modify other registers, as they default to the values that comply with the USB Battery Charging Specification.
4. Configure timing parameters to obtain flexibility as per the requirements of a specific system.

All module configuration must occur *before* initiating the charger detection sequence. Configuration changes made *after* setting CONTROL[START] result in undefined behavior.

63.6 Application information

This section provides application information.

63.6.1 External pullups

Any external pullups applied to the USB D+ or D- data lines must be capable of being disabled to prevent incorrect pullup values or incorrect operation of the USB subsystem.

63.6.2 Dead or weak battery

According to the *USB Battery Charging Specification, Revision 1.2*, a USB device with a dead or weak battery that is attached to an SDP can draw charging current from VBUS without connecting for several minutes, until the battery is charged to the point that the USB device can connect. The conditions for this operation are referred to as the Dead Battery Provision.

The *USB Battery Charging Specification, Revision 1.2* limits this charging condition to 100 mA for a duration of 45 minutes. The later *USB 2.0 Connect Timing Update ECN* modifies this charging condition to 500 mA for a duration of 2 minutes. Customers designing systems to take advantage of the Dead Battery Provision should study both specifications closely.

The USBDCD module is compatible with systems that do not check the strength of the battery. Therefore, this module assumes that the battery is good, so the USB device must immediately connect to the USB bus by pulling the USB_DP pin high after the USBDCD module has determined that the device is attached to an SDP or CDP. (By definition, a DCP does not support connection but still requires a signaling event, see DCP.)

The module is also compatible with systems that have other circuitry to check the strength of the battery. In these systems, if it is known that the battery is weak or dead, software can delay connecting to the USB while charging using the Dead Battery Provision. Once the battery is charged to the good battery threshold, software must then connect to the USB host by pulling the USB_DP pin high.

While using the Dead Battery Provision, the USB_DP pin must signal by enabling V_{DP_SRC}. On this product that signaling can be done most simply by setting the SIGNAL_OVERRIDE[PS] field to value 10b.

63.6.3 Handling unplug events

If the device is unplugged from the USB bus during the charger detection sequence, the contents of the STATUS register must be ignored and the USBDCD module must get a Software Reset, as described in Software reset.

63.7 USBDCD register descriptions

63.7.1 USBDCD memory map

USBDCD0 base address: 400D_C000h

USBHS1_PHY_DCD base address: 4010_A800h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CONTROL)	32	RW	0003_0000h
4h	Clock (CLOCK)	32	RW	0000_0061h
8h	Status (STATUS)	32	R	0000_0000h
Ch	Signal Override (SIGNAL_OVERRIDE)	32	RW	0000_0000h
10h	TIMER0 (TIMER0)	32	RW	0010_0000h
14h	TIMER1 (TIMER1)	32	RW	000A_0028h
18h	TIMER2_BC11 (TIMER2_BC11)	32	RW	0028_0001h
18h	TIMER2_BC12 (TIMER2_BC12)	32	RW	0001_0028h

63.7.2 Control (CONTROL)

Offset

Register	Offset
CONTROL	0h

Function

Controls various global functions, such as software reset, initiating the charger detection sequence, BC12 compatibility, and interrupt enablement.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						0	0	0						BC12	IE
W							SR	START								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							IF	0							0
W																IACK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25 SR	Software Reset Determines whether a software reset is performed. 0b - Do not perform a software reset. 1b - Perform a software reset.
24 START	Start Change Detection Sequence Determines whether the charger detection sequence is initiated. 0b - Do not start the sequence. Writes of this value have no effect. 1b - Initiate the charger detection sequence. If the sequence is already running, writes of this value have no effect.
23-18 —	Reserved
17 BC12	Battery Charging Revision 1.2 Compatibility BC1.2 compatibility. This field cannot be changed after start detection. 0b - Compatible with BC1.1 1b - Compatible with BC1.2 (default)
16 IE	Interrupt Enable Enables/disables interrupts to the system. 0b - Disable interrupts to the system. 1b - Enable interrupts to the system.
15-9 —	Reserved
8 IF	Interrupt Flag Determines whether an interrupt is pending. 0b - No interrupt is pending. 1b - An interrupt is pending.
7-1 —	Reserved
0 IACK	Interrupt Acknowledge Determines whether the interrupt is cleared.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Do not clear the interrupt. 1b - Clear the IF field (interrupt flag).

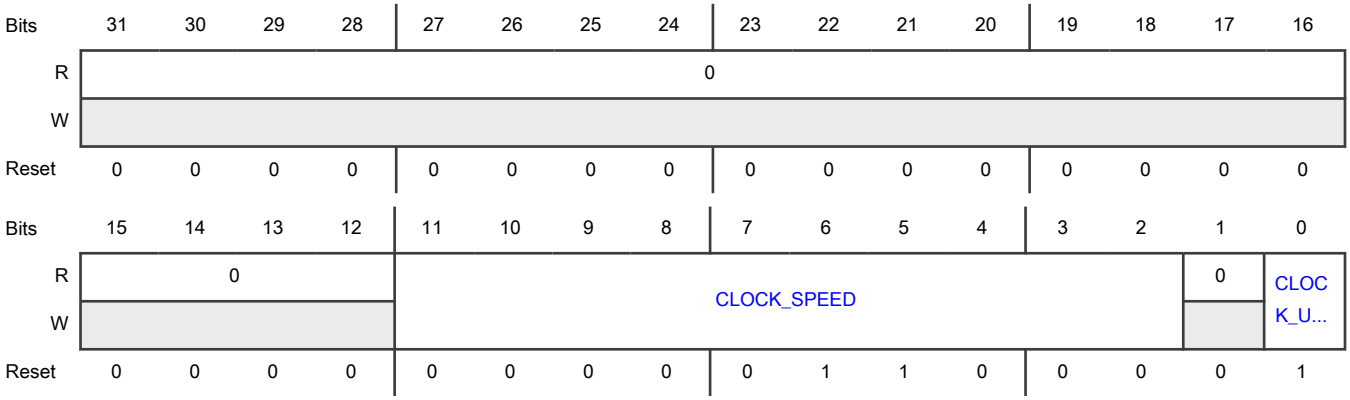
63.7.3 Clock (CLOCK)

Offset

Register	Offset
CLOCK	4h

Function
Controls the clock speed.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-2 CLOCK_SPEED	Numerical Value of Clock Speed in Binary The unit of measure is programmed in CLOCK_UNIT. The valid range is from 1 to 1023 when the clock unit is MHz and 4 to 1023 when the clock unit is kHz. Examples with CLOCK_UNIT = 1b: • For 48 MHz: 00110000b (48) • For 24 MHz: 00011000b (24) Examples with CLOCK_UNIT = 0b:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• For 100 kHz: 01100100b (100) • For 500 kHz: 000111110100b (500)
1 —	Reserved
0 CLOCK_UNIT	Unit of Measurement Encoding for Clock Speed Specifies the unit of measure for the clock speed. 0b - kHz Speed (between 4 kHz and 1023 kHz) 1b - MHz Speed (between 1 MHz and 1023 MHz)

63.7.4 Status (STATUS)

Offset

Register	Offset
STATUS	8h

Function

Stores the current state of the module for system software monitoring.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0									ACTIVE	TO	ERR	SEQ_STAT		SEQ_RES	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-23	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
22 ACTIVE	Active Status Indicator Indicates whether the sequence is running. 0b - The sequence is not running. 1b - The sequence is running.
21 TO	Timeout Flag Indicates whether the detection sequence is passed the timeout threshold. 0b - The detection sequence is not running for over 1 s. 1b - It is over 1 s since the data pin contact was detected and debounced.
20 ERR	Error Flag Indicates whether there is an error in the detection sequence. 0b - No sequence errors. 1b - Error in the detection sequence. See STATUS[SEQ_STAT]to determine the phase in which the error occurred.
19-18 SEQ_STAT	Charger Detection Sequence Status Indicates the status of the charger detection sequence. 00b - The module is either not enabled, or the module is enabled but the data pins have not yet been detected. 01b - Data pin contact detection is complete. 10b - Charging port detection is complete. 11b - Charger type detection is complete.
17-16 SEQ_RES	Charger Detection Sequence Results Reports how the charger detection is attached. 00b - No results to report. 01b - Attached to an SDP. Must comply with USB 2.0 by drawing only 2.5 mA (max) until connected. 10b - Attached to a charging port. The exact meaning depends on the STATUS[SEQ_STAT] field (value 0: Attached to either a CDP or a DCP. The charger type detection has not completed. value 1: Attached to a CDP. The charger type detection has completed.) 11b - Attached to a DCP.
15-0 —	Reserved **NOTE** Fields do not always read as 0.

63.7.5 Signal Override (SIGNAL_OVERRIDE)

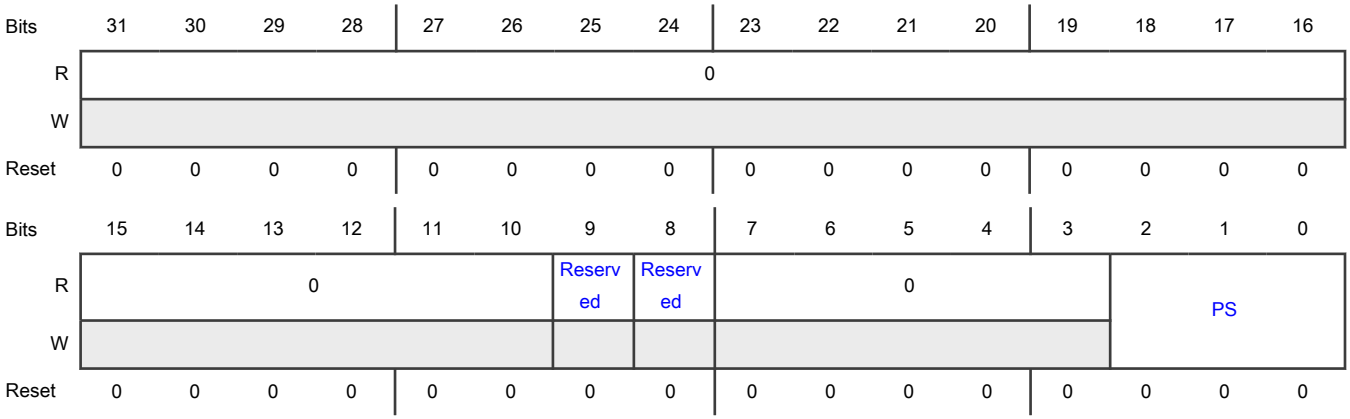
Offset

Register	Offset
SIGNAL_OVERRIDE	Ch

Function

Provides a way for the customer to enable signaling required by the USB BC Rev. 1.2 Specification after the battery charger detection sequences is completed.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 —	Reserved Reserved, not for customer use. The value of this field may change during module operation.
8 —	Reserved Reserved, not for customer use. The value of this field may change during module operation.
7-3 —	Reserved
2-0 PS	Phase Selection Enables specified voltage and current source circuits on the USB_DP and USB_DM pins. Use of the override values below enables the charger detection bias circuits without needing to set CONTROL[START].

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Customers may set this field to 010b for required signaling if attached to a Dedicated Charging Port, or during operation under the Dead Battery Provision. Customers may set this field to 100b in Embedded Host use case for signaling Charging Downstream Port advertisement. 000b - No overrides. Field must remain at this value during normal USB data communication to prevent unexpected conditions on USB_DP and USB_DM pins. (Default) 001b - Reserved, not for customer use. 010b - Enables VDP_SRC voltage source for the USB_DP pin and IDM_SINK current source for the USB_DM pin. 011b - Reserved, not for customer use. 100b - Enables VDM_SRC voltage source only. 101b - Reserved, not for customer use. 110b - Reserved, not for customer use. 111b - Reserved, not for customer use.

63.7.6 TIMER0 (TIMER0)

Offset

Register	Offset
TIMER0	10h

Function
Represents the system latency in ms using the TSEQ_INIT field.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								TSEQ_INIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				TUNITCON											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25-16 TSEQ_INIT	Sequence Initiation Time TSEQ_INIT represents the system latency (in ms) measured from the time VBUS goes active to the time system software initiates the charger detection sequence in the USBDCD module. When software sets CONTROL[START], the Unit Connection Timer (TUNITCON) is initialized with the value of TSEQ_INIT. Valid values are 0-1023, but the USB Battery Charging Specification requires the entire sequence, including TSEQ_INIT, to be completed in 1 s or less. 00_0000_0000b-11_1111_1111b - 0 ms - 1023 ms
15-12 —	Reserved
11-0 TUNITCON	Unit Connection Timer Elapse (in ms) Displays the amount of elapsed time since the event of setting CONTROL[START] plus the value of TSEQ_INIT. The timer is automatically initialized with the value of TSEQ_INIT before starting to count. This timer enables compliance with the maximum time allowed to connect T_{UNIT_CON} under the USB Battery Charging Specification. If the timer reaches the one second limit, the module triggers an interrupt and sets the error flag STATUS[ERR]. The timer continues counting throughout the charger detection sequence, even when control has been passed to software. As long as the module is active, the timer continues to count until it reaches the maximum value of FFFh (4095 ms). The timer does not roll over to zero. A software reset clears the timer.

63.7.7 TIMER1 (TIMER1)

Offset

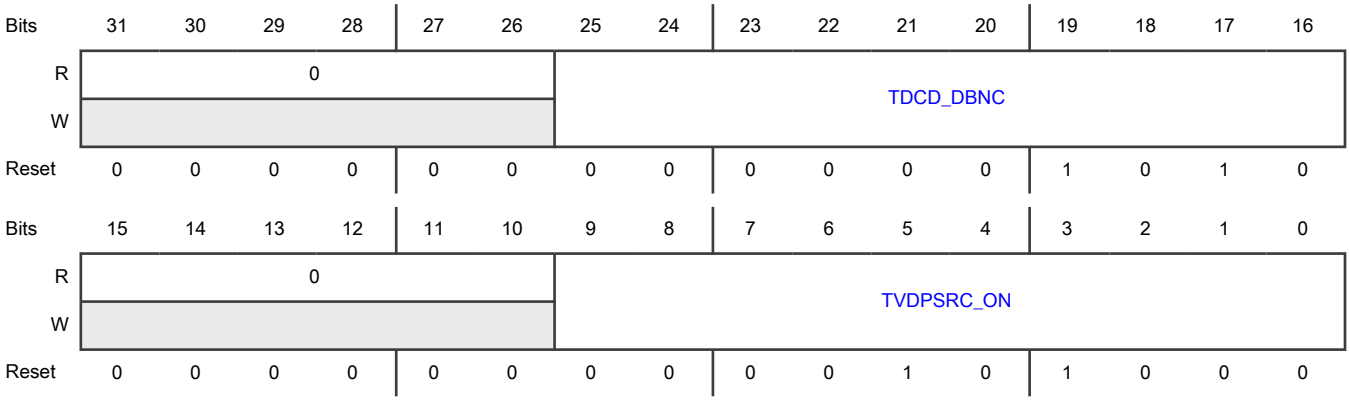
Register	Offset
TIMER1	14h

Function
Contains timing parameters.

NOTE

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 TDCD_DBNC	Time Period to Debounce D+ Signal Sets the time period (ms) to debounce the D+ signal during the data pin contact detection phase. See Debouncing the data pin contact. Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 10 ms. 00_0000_0001b-11_1111_1111b - 1 ms - 1023 ms
15-10 —	Reserved
9-0 TVDPSRC_ON	Time Period Comparator Enabled This timing parameter is used after detection of the data pin. See Charging port detection. Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 40 ms. 00_0000_0001b-11_1111_1111b - 1 ms - 1023 ms

63.7.8 TIMER2_BC11 (TIMER2_BC11)

Offset

Register	Offset
TIMER2_BC11	18h

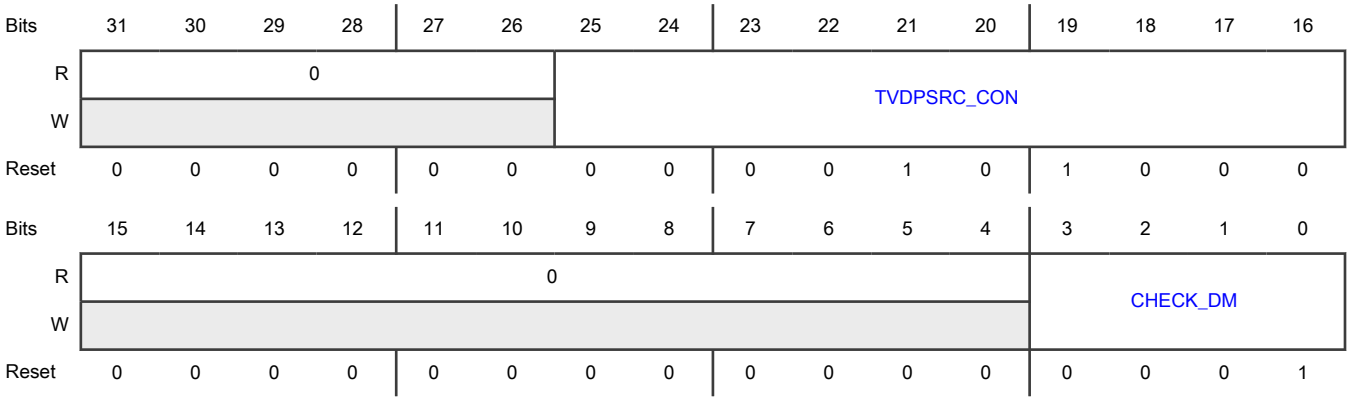
Function

TIMER2_BC11 contains timing parameters for *USB Battery Charging Specification, Rev 1.1.*

NOTE

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 TVDPSRC_CON	Time Period Before Enabling D+ Pullup Sets the time period (ms) that the module waits after charging port detection before system software must enable the D+ pullup to connect to the USB host. Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 40 ms. 00_0000_0001b-11_1111_1111b - 1 ms - 1023 ms
15-4 —	Reserved
3-0 CHECK_DM	Time Before Check of D- Line Sets the amount of time (in ms) that the module waits after the device connects to the USB bus until checking the state of the D- line to determine the type of charging port. See Charger type detection. Valid values are 1-15 ms. 0001b-1111b - 1 ms - 15 ms

63.7.9 TIMER2_BC12 (TIMER2_BC12)

Offset

Register	Offset
TIMER2_BC12	18h

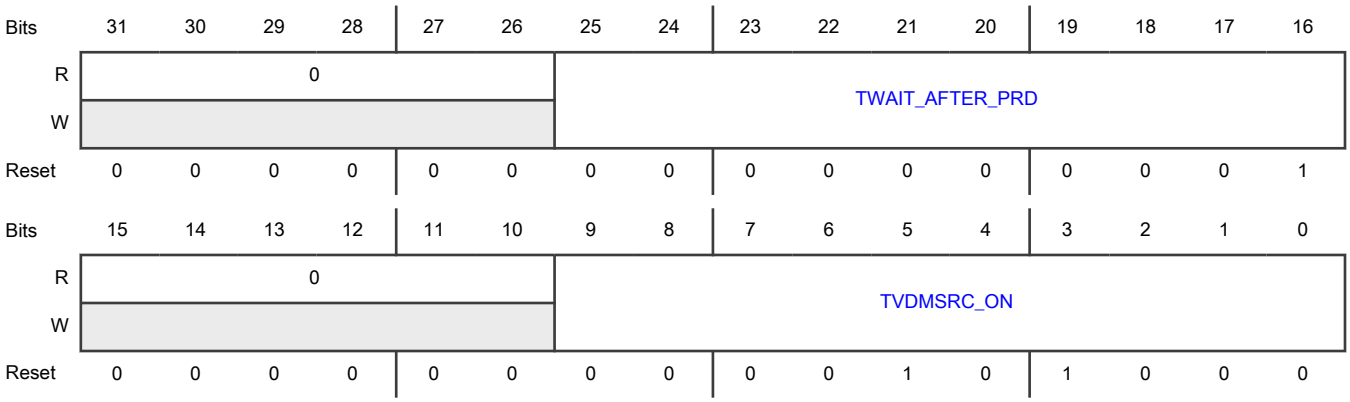
Function

Contains timing parameters for *USB Battery Charging Specification, Rev 1.2*.

NOTE

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 TWAIT_AFTER_PRD	TWAIT_AFTER_PRD Sets the amount of time (in ms) that the module waits after primary detection before start to secondary detection. Valid values are 1-1023ms. 00_0000_0001b-11_1111_1111b - 1 ms - 1023 ms
15-10 —	Reserved
9-0 TVDMSRC_ON	TVDMSRC_ON Sets the amount of time (in ms) that the module enables the V_{DM_SRC}. Valid values are 0-40ms. 00_0000_0000b-00_0010_1000b - 0 ms - 40 ms

Chapter 64
Universal Serial Bus High-Speed Controller (USBHS)

64.1 Chip-specific USBHS information

Table 507. Reference links to related information

Topic	Related module	Reference
Full description	USBHS	USBHS
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

NOTE

See AHB Clock Control 2 (AHBCLKCTRL2) for clock control. Also see Peripheral Reset Control 2 (PRESETCTRL2) for reset control.

64.1.1 Module instances

This device has one instance of the USBHS module.

64.1.2 USBHS operation modes

USBHS is functional only in SD and OD modes. It is non-functional in MD mode. Configure SPC.ACTIVE_CFG[DCDC_VDD_LVL] = SPC.ACTIVE_CFG[CORELDO_VDD_LVL] >= 0x2 for correct operation of the module.

64.2 Overview

The USB controller block provides high performance USB functionality that conforms to the *Universal Serial Bus Specification*, Rev. 2.0 (Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips; 2000), and the *On-The-Go and Embedded Host Supplement to the USB Revision* 2.0 Specification (Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, Renesas Electronics Corporation, ST-Ericsson; 2012).

The USB controller consists of one independent USB controller core: On-The-Go (OTG) controller core. Each controller core supports UTMI interface. See Features for more details. The controller core is single-port core. For the OTG core, there is only one port. The port can be used as either a downstream or an upstream port.

64.2.1 Block diagram

The following figure is a block diagram of USB.

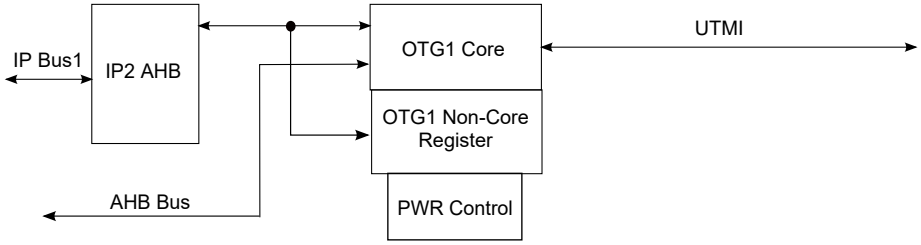


Figure 348. USB block diagram

64.2.2 Features

There is one USB 2.0 controller core in this chip:

- Controller Core 0 is also named 'OTG1 Core'; its connected port is named 'OTG1 port'.

The following list provides features of each controller core.

- USB 2.0 Controller Core 0
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface
  - High Speed, Full Speed and Low Speed operation in HOST mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, session request protocol, and host negotiation protocol
  - 8 bidirectional endpoints
  - Support charger detection
- Low-power mode with local and remote wake-up capability
- Embedded DMA controller in each core

64.3 Functional description

These sections describe the functionality of the various building blocks of the USB.

64.3.1 USB 2.0 Controller Core 0

The USB 2.0 Controller 0 is an instantiation of an EHCI-compatible core which supports high-, full-, and low-speed operation.

In Host mode, this controller core supports high-, full-, and low-speed operation. In Device mode, it supports high- and full-speed operation.

64.3.1.1 Host mode

The controller supports direct connection of a HS/FS/LS device with on-chip UTMI transceiver. Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and protocol engine blocks to support connection to full and low speed devices.

64.3.1.2 Peripheral (device) mode

- 8 bidirectional endpoints

- High/full-speed operation
- Support of HNP, and SRP
- Remote wake-up capability

64.3.2 Modes of operation

The USB has two main modes of operation: normal mode and low power mode .

Each USB OTG controller core can operate in High Speed operation (480 Mbps), Full Speed operation (12 Mbps) and Low Speed operation (1.5 Mbps).

This chapter explains the operation modes.

64.3.2.1 Normal mode

The OTG controller core can operate in Host mode and Device (Peripheral) mode.

The USB controller core has its corresponding port, which can work in one or more interface modes.

NOTE

The USB controller supports only the interface type listed below. Selecting a different interface type in the PORTSC_PTS field results in unpredictable behavior and may cause the system to hang.

- OTG1 port
  - This port supports on-chip UTMI transceiver only.

64.3.2.2 Low-power mode

The USB controller core has a low-power mode (Suspend mode) to save power consumption.

As described in the USB 2.0 specification, the device can go into the Suspend state after it sees a constant Idle state on the upstream facing port. The OTG controller core enters the Suspend mode after 3 ms of inactivity on the port when it is in Device Operation mode. The host controllers, including the OTG controller in Host mode, do not suspend automatically but can be placed in the Suspend mode by software.

Either the local Arm platform or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication. For details about Suspend/Resume, see Power control.

64.3.2.2.1 Power control

The USB controller supports suspend and wake-up functionality.

The power control block allows for placing the transceiver in USB low power mode when USB bus is IDLE, and supports local and remote wake-up to bring the transceiver out of USB low power mode when needed. Additionally, the power control block can wake-up the Arm platform from core sleep mode by generating an interrupt.

64.3.2.2.1.1 Entering low power suspend mode

In Host operation mode, low power suspend mode is entered as follows:

1. Clear the ASE and PSE bits in USB_USBCMD, and wait until the AS and PS bits in USB_USBSTS become "0".
2. Set the "SUSPEND" bit in USB_PORTSC1
3. Set VBUSVALID_TO_SESSVALID in USBPHYx_USB1_VBUS_DETECT
4. Set the "PHCD" bit in USB_PORTSC1
5. Set all PWD bits in USBPHYx_PWD
6. Set CLKGATE in USBPHYx_CTRL

NOTE

Step 4,5,6 shall be done in atomic operation. That is, interrupt should be disabled during these three steps.

For device operation mode, low power suspend mode is entered as follows:

1. After Host drive is IDLE for 3ms, an SLI interrupt is issued (the "DCSUSPEND" or "SLI" bit in USB_USBSTS)
2. Set VBUSVALID_TO_SESSVALID in USBPHYx_USB1_VBUS_DETECT
3. Set the "PHCD" bit on USB_PORTSC1
4. Set all PWD bits in USBPHYx_PWD
5. Set CLKGATE in USBPHYx_CTRL

NOTE

Step 3,4,5 shall be done in atomic operation. That is, interrupt should be disabled during these three steps.

64.3.2.2.1.2 Wake-up events

The power control block monitors the USB bus when the USB core is in the USB suspend state.

Depending on whether the core is on Host or Device mode, a number of wake-up conditions are monitored. Upon detection of a wake-up condition, an interrupt will be generated to Arm platform if the related wake-up interrupt enable bit is set.

USB wake-up interrupt also re-activates the Arm platform clocks if they were stopped during the suspend.

64.3.2.2.1.2.1 Host mode events

The host controller wakes up on the following events:

- Remote Wake-up Request

A peripheral can request the host to reactivate the bus by driving wake-up signaling on the DM/DP lines. The power control block sends a wake-up request to the USB core when a J-K transition on DM/DP line is detected.

- Wake-Up On Overcurrent

If Wake-Up On Overcurrent is enabled (WKOC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when an overcurrent event is detected.

- Wake-Up On Disconnect

If Wake-Up On Disconnect is enabled (WKDC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when a disconnection event is detected (J-SE0/K-SE0 transition on DM/DP line).

- Wake-Up On Connect

If a Wake-Up On Connect is enabled (WKCN bit in the USB core register PORTSC1 is set '1'), the power control sub-block sends a wake-up request to the USB core when the connection event is detected (SE0-J/SE0-K transition on DM/DP line).

For a detailed description of register bits WKOC, WKDC, WKCN, see USB_nPORTSC1.

64.3.3 Interrupts

64.3.3.1 USB core interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB Cores. Refer to the USB_nUSBINTR for details.

64.3.3.2 USB wake-up interrupts

Each USB Core has an associated wake-up interrupt. The wake-up interrupts are generated outside of the USB controller cores, but using the same vector as the corresponding USB controller cores interrupt.

These interrupts are generated by the Power Control blocks which run on the 32 kHz standby clock. The wake-up interrupt is designed to work even when the USB and Arm platform clocks are disabled, such that a wake-up condition on the USB bus can re-activate the Arm platform clocks.

Because the wake-up interrupt is generated and cleared on a 32 kHz clock, this interrupt request responds very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt masks the request instantaneously as this is clocked by the Arm platform clock. The software should wait for at least three 32 kHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. Because this interrupt is only used during low power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to entering the USB suspend mode.

64.3.3.3 Interrupts - host operational model

For detailed information about interrupt capability that the EHCI host controller hardware provides, see the "Interrupts" section in *Enhanced Host Controller Interface (EHCI) Specification for Universal Serial Bus, revision 1.0*.

64.3.3.4 Servicing interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

64.3.3.4.1 High-frequency interrupts

High frequency interrupts in particular should be handled in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

The table below describes the High frequency interrupt events.

Table 508. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt - USB.ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Figure 352 shows the End Point Queue Head). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt[1] - USB.ENDPTCOMPLETE	Handle completion of dTD as indicated in Figure 352 shows the End Point Queue Head.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1. It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

64.3.3.4.2 Low-frequency interrupts

The low frequency interrupts can be handled in any order because they do not occur often in comparison to the high-frequency interrupts.

The table below shows the Low frequency interrupt events.

Table 509. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

64.3.3.4.3 Error interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

The following table shows the error interrupt events.

Table 510. Error Interrupt Events

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ USB.ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

64.3.4 Clocks

The following table describes the clock sources for USB. See the clock control chapter for clock setting, configuration and gating information.

Table 511. USB Clocks

Clock	Description
ipg_ahb_clk	AHB bus clock
ipg_clk_s	Peripheral access clock

64.4 External signals

The table below lists the external signals of USB.

Table 512. USB External Signals

Signal	Description	Direction
OTGn_DN	DN OTG Signal	I/O
OTGn_DP	DP OTG Signal	I/O
OTGn_ID	ID signal	I/O
OTGn_OC	OTG External input for VBUS overcurrent detection	I

Table continues on the next page...

Table 512. USB External Signals (continued)

Signal	Description	Direction
OTGn_PWR	To control PMIC to supply VBUS voltage	O

64.5 Initialization

64.5.1 Host controller initialization

For detailed information, see the "Host Controller Initialization" section in *Enhanced Host Controller Interface (EHCI) Specification for Universal Serial Bus, revision 1.0*.

64.5.2 Device controller initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs.

Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

- Set Controller Mode in the USB.USBMODE register to device mode.

  NOTE

  Transitioning from host mode to device mode requires a device controller reset before modifying USB.USBMODE.

- Allocate and Initialize device queue heads in system memory.
  - Minimum: Initialize device queue heads 0 Tx & 0 Rx.

    NOTE

    All device queue heads for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

  - For information on device queue heads, refer to section Device data structures.
- Configure USB.ENDPOINTLISTADDR Pointer.
  - For additional information on USB.ENDPOINTLISTADDR, refer to the register table.
- Enable the microprocessor interrupt associated with the USB core.
  - Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
  - For a list of available interrupts refer to the USB_nUSBINTR and the USB_nUSBSTS register tables.
- Set Run/Stop bit to Run Mode.
  - After the Run bit is set and the device is connected to a host, a Bus Reset will be issued by host downstream port. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the Port State and Control section below.

    NOTE

    Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to prime Endpoint 0 initially because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

64.6 Application information

This section describes the detailed application knowledge for OTG1 port.

64.6.1 Register interface

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

NOTE

USB controller registers support only DWORD (32-bit) access.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are dynamic control or status registers that may be read only, read/write, or read/write-to-clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

The following table describes the Interface register sets.

Table 513. Interface Register Sets

Offset	Register Set	Explanation
000h-07Ch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h-124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
080h-0FCh 140h-1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

64.6.1.1 Configuration, control and status register set

The following table describes the Device/Host capability registers.

NOTE

Depending on implementation, "*x*" can have the following values: UOG1.

Table 514. Device/Host Capability Registers

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
000h	4	USB_*x*_ID	Identification Register	O	O
004h	4	USB_*x*_HWGENERAL	General Hardware Parameters	O	O
008h	4	USB_*x*_HWHOST	Host Hardware Parameters	X	O
00Ch	4	USB_*x*_HWDEVICE	Device Hardware Parameters	O	X

Table continues on the next page...

Table 514. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
010h	4	USB_*x*_HWTXBUF	TX Buffer Hardware Parameters	O	O
014h	4	USB_*x*_HWRXBUF	RX Buffer Hardware Parameters	O	O
018-07Fh		-	Reserved		
080h	4	USB_*x*_GPTIMER0LD	General Purpose Timer #0 Load Register	O	O
084h	4	USB_*x*_GPTIMER0CTRL	General Purpose Timer #0 Control Register	O	O
088h	4	USB_*x*_GPTIMER1LD	General Purpose Timer #1 Load Register	O	O
08Ch	4	USB_*x*_GPTIMER1CTRL	General Purpose Timer #1 Control Register	O	O
090h	4	USB_*x*_SBUSCFG	System Bus Interface Configuration Register	O	O
094-09Fh		-	Reserved		
100h	1	USB_*x*_CAPLENGTH	Capability Register Length	O	O
101h		-	Reserved		
102h	2	USB_*x*_HCIVERSION	Host Controller Interface Version Number	X	O
104h	4	USB_*x*_HCSPARAMS	Host Controller Structural Parameters	X	O
108h	4	USB_*x*_HCCPARAMS	Host Controller Capability Parameters	X	O
10C-11Fh		-	Reserved		
120h	2	USB_*x*_DCIVERSION	Device Controller Interface Version Number	O	X
122h	2	-	Reserved		
124h	4	USB_*x*_DCCPARAMS	Device Controller Capability Parameters	O	X
128-13Fh		-	Reserved		
140h	4	USB_*x*_USBCMD	USB Command Register	O	O
144h	4	USB_*x*_USBSTS	USB Status Register	O	O
148h	4	USB_*x*_USBINTR	USB Interrupt Enable Register	O	O
14Ch	4	USB_*x*_FRINDEX	USB Frame Index	O	O
150h	4	-	Reserved		

Table continues on the next page...

Table 514. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
154h	4	USB_*x*_PERIODICLISTBASE	Frame List Base Address	X	O
		USB_*x*_DEVICEADDR	USB Device Address	O	X
158h	4	USB_*x*_ASYNCLISTADDR	Next Asynchronous List Address	X	O
	4	USB_*x*_ENDPOINTLISTADDR	Address at Endpoint list in memory	O	X
15Ch	4	-	Reserved		
160h	4	USB_*x*_BURSTSIZE	Programmable Burst Size	O	O
164h	4	USB_*x*_TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning	X	O
168h	4	-	Reserved		
170h	4	-	Reserved		
178h	4	USB_*x*_ENDPTNAK	Endpoint NAK register	O	X
17Ch	4	USB_*x*_ENDPTNAKEN	Endpoint NAK Enable register	O	X
180h	4	USB_*x*_CONFIGFLAG	Configured Flag Register	X	O
184h	4	USB_*x*_PORTSC1	Port Status/Control Register 1	O	O
188-1A3h		-	Reserved		
1A4h	4	USB_*x*_OTGSC	On-The-Go Status/Control Register (OTG only)	O	O
1A8h	4	USB_*x*_USBMODE	USB Controller Operating Mode	O	O
1ACh	4	USB_*x*_ENDPTSETUPSTAT	Endpoint Setup Status	O	X
1B0h	4	USB_*x*_ENDPTPRIME	Endpoint Initialization	O	X
1B4h	4	USB_*x*_ENDPTFLUSH	Endpoint De-Initialization	O	X
1B8h	4	USB_*x*_ENDPTSTATUS	Endpoint Status	O	X
1BCh	4	USB_*x*_ENDPTCOMPLETE	Endpoint Complete	O	X
1C0	64	USB_*x*_ENDPTCTRL0	Endpoint Control Register 0-7	O	X

Table continues on the next page...

Table 514. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
1C4 ... 1DCh		USB_*x*_ENDPTCTRL1 .... USB_*x*_ENDPTCTRL7			

NOTE

"O" means the register is available in host/device operation mode;

"X" means the register is reserved in host/device operation mode

64.6.1.2 Identification registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

64.6.1.3 OTG operations

64.6.1.3.1 Register bits

In the previous section, the Register interface has behaviors described for device mode and behaviors described for host mode. However, for OTG operations it is necessary to perform tasks independent of the controller mode.

NOTE

The only way to transit the controller mode out of host or device mode is with the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

The following figure shows the controller mode.

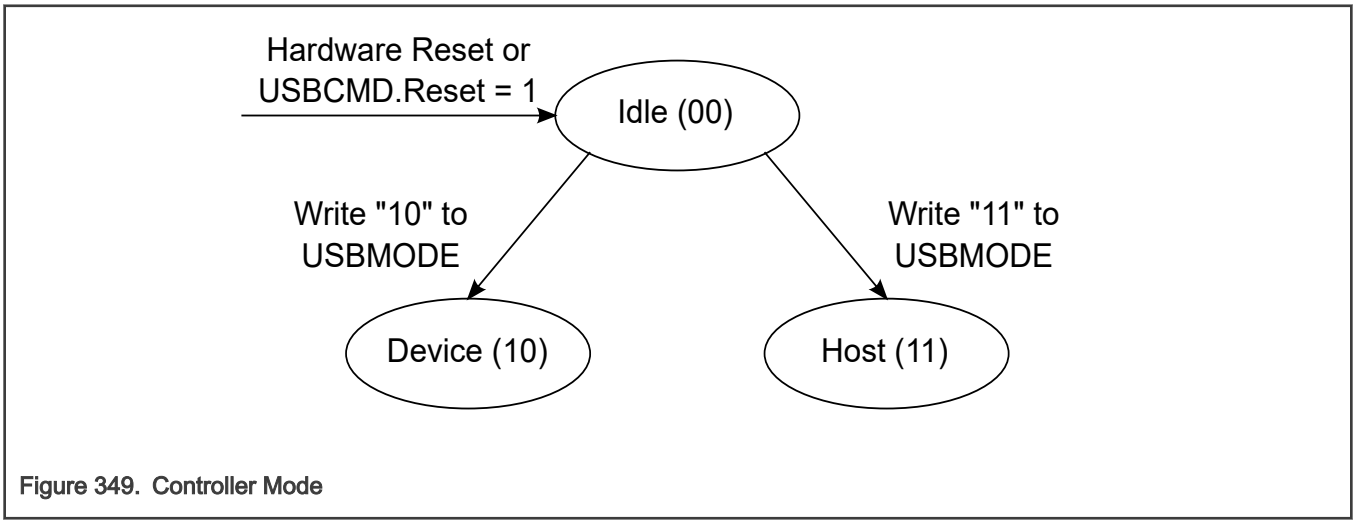


Figure 349. Controller Mode

To this end, listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

All Identification Registers

All Device/Host Capability Registers

OTGSC: All bits

PORTSC1:

- Physical Interface Select
- Physical Interface Serial Select
- Physical Interface Data Width
- Physical Interface Low Power
- Physical Interface Wake Signals
- Port Indicators
- Port Power

64.6.2 Host data structures

For detailed information, see the "Data Structures" chapter in *Enhanced Host Controller Interface (EHCI) Specification for Universal Serial Bus, revision 1.0*.

64.6.3 Host operational model

For detailed information, see the "Operational Model" chapter in *Enhanced Host Controller Interface (EHCI) Specification for Universal Serial Bus, revision 1.0*.

64.6.4 EHCI deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specification. Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary.

The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator - Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

64.6.4.1 Embedded transaction translator function

The OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

64.6.4.1.1 Discovery-EHCI deviation

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation.

The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, Chirp completes successfully).

Because this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in USB.PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in the following table.

Table 515. Summary of EHCI

Standard EHCI	EHCI with embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (i.e. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (i.e. Split target hub is the root hub)]

64.6.4.1.2 Operational model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specification. Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Because the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

64.6.4.1.2.1 Micro- frame pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

64.6.4.1.2.2 Split state machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator.

Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. The following table summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 516. Summary of the Conditions of Handshakes[1]

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (OK)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

1. The unshaded cells represent Start-Splits and the shaded cells represent Complete-Splits

64.6.4.1.2.3 Asynchronous transaction scheduling and buffer management

USB 2.0 specification items	Implementation in the embedded transaction translator
11.17.3	Sequencing is provided and a packet length estimator ensures that no FS or LS packet babbles exist in the SOF time
11.17.4	Transaction tracking happens for two data pipes

64.6.4.1.2.4 Periodic transaction scheduling and buffer management

USB 2.0 specification items	Implementation in the embedded transaction translator
11.18.6.[1-2]	• Abort of pending start-split conditions: — End of frame (EOF), and not started in six microframes — Idle for more than four microframes • Abort of pending complete-split conditions:

Table continues on the next page...

Table continued from the previous page...

USB 2.0 specification items	Implementation in the embedded transaction translator
	— EOF — Idle for more than four microframes
11.18.[7-8]	• Transaction tracking for up to 16 data pipes • Complete-split transaction searching **NOTE** There is no data schedule mechanism for these transactions other than the microframe pipeline. The embedded TT assumes that the number of packets scheduled in a frame does not exceed the frame duration (1 ms); else, it results in an undefined behavior.

64.6.4.1.2.5 Multiple transaction translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the HCSPARAMS register.

64.6.4.2 Device operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

64.6.4.2.1 USB_USBMODE register

Given that the dual-role controller is initialized in neither host nor device mode, the USB_nUSBMODE register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

64.6.4.2.2 Non-zero fields the register file

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

64.6.4.2.3 SOF interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode.

EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USB_nUSBSTS and USB_nUSBINTR registers.

64.6.4.3 Embedded design interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

64.6.4.3.1 Frame adjust register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. That is, a 60 MHz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 MHz transceiver clock for 16-bit physical interfaces.

64.6.4.4 Miscellaneous variations from EHCI

64.6.4.4.1 Programmable physical interface behaviour

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the USB_nPORTSC1 register providing a capability that is not defined by EHCI.

64.6.4.4.2 Discovery

64.6.4.4.2.1 Port reset

The port connect methods specified by EHCI require setting the port reset bit in the USB_nPORTSC1 register for a duration of 10 ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10 ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to reset the device.
- Software shall write a '0' to reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress, the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

64.6.4.4.2.2 Port speed detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation, which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices.

Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed - *This information is redundant with the 2-bit Port Speed indicator above.*

64.6.4.4.3 Port test mode

Port Test Control mode behaves fully as described in EHCI. An alternate host controller driver procedure is not necessary or supported.

64.6.5 Device data structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller.

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

NOTE

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writable fields. The device controller must preserve the read-only fields on all data structure writes.

The figure below shows the organization of the EndPoint Queue Head.

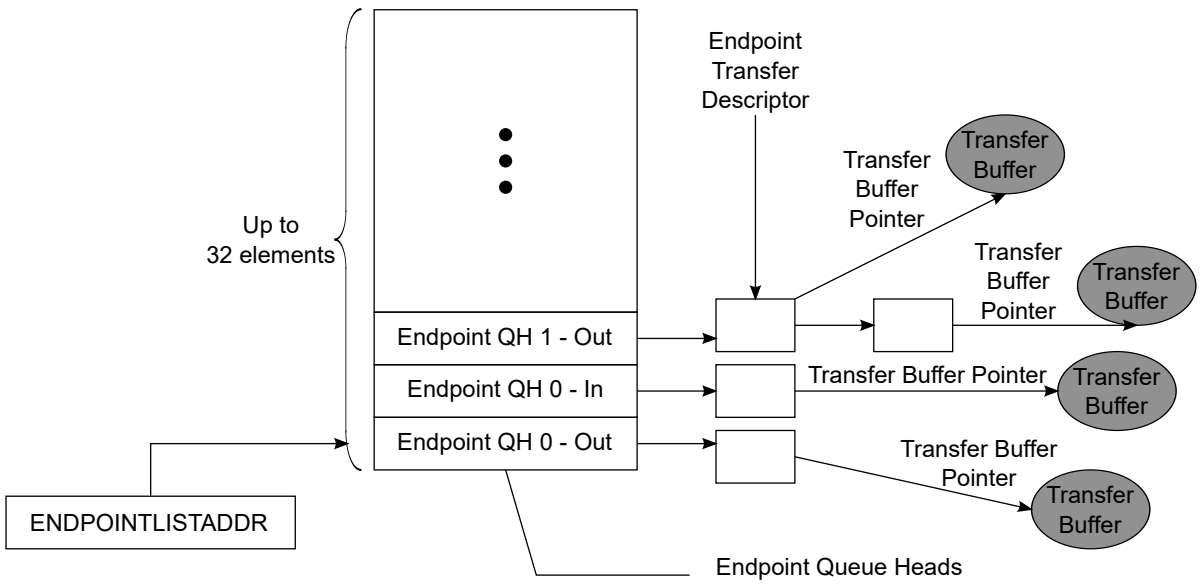


Figure 350. EndPoint Queue Head Organization

Endpoint queue heads are arranged in an array in a continuous area of memory pointed to by the USB.ENDPOINTLISTADDR pointer. The even-numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). That is, the device queue heads in the list with even offset support receiver endpoints (OUT) and the queue heads in the list with odd offset are used for transmit endpoint(IN). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE

The Endpoint Queue Head List must be aligned to a 2k boundary.

64.6.5.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers for a given endpoint are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries.

During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status

DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

Table 517. Endpoint Queue Head (dQH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult		zlt	0		Maximum Packet Length											ios	0														
Current dTD Pointer																											0				
Next dTD Pointer																											0				T [1]
0	Total Bytes															ioc	0			MultO	0	Status									
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				Reserved											
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											
Buffer Pointer (Page 4)[1]																				Reserved											
Reserved																															
Set-up Buffer Bytes 3…0																															
Set-up Buffer Bytes 7…4																															

1. Transfer overlay starts at T and continues through Buffer Pointer (Page 4).

Host Controller Read/Write Host Controller Read Only

64.6.5.1.1 Endpoint capabilities/characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 518 describes the endpoint capabilities.

Table 518. Endpoint Capabilities/Characteristics

Bit	Description
31-30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 - Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. **NOTE** Non-ISO endpoints must set Mult="00". ISO endpoints must set Mult="01", "10", or "11" as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0 - Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 - Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28-27	Reserved. These bit reserved for future use and should be set to zero.
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14-0	Reserved. Bits reserved for future use and should be set to zero.

64.6.5.1.2 Transfer overlay-endpoint queue head

The seven DWords in the overlay area represent a transaction working space for the device controller.

The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

64.6.5.1.3 Current dTD pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

The following table describes the dTD Pointer.

Table 519. Next dTD Pointer

Bit	Description

Table continues on the next page...

Table 519. Next dTD Pointer (continued)

31-5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4-0	Reserved. Bit reserved for future use and should be set to zero.

64.6.5.1.4 Set-up buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

The following table describes the Multiple Mode Control.

Table 520. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31-0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31-0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

64.6.5.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/received for a given transfer.

The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing transfers with transfer descriptors.

Table below shows the Endpoint Transfer Descriptor (dTD).

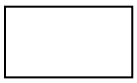
Table 521. Endpoint Transfer Descriptor (dTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Link Pointer																											0				T
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				0	Frame Number										
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											

Table continues on the next page...

Table 521. Endpoint Transfer Descriptor (dTD) (continued)

Buffer Pointer (Page 4)	Reserved

Host Controller Read/Write  Host Controller Read Only

The following table describes the dTD Pointer.

Table 522. Next dTD Pointer

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

The following table describes the dTD Token.

Table 523. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30-16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K (5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K (4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14-12	Reserved. Bits reserved for future use and should be set to zero.
11-10	Multiplier Override (MultO). This field can be used for transmit ISO's (i.e. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example:

Table continues on the next page...

Table 523. dTD Token (continued)

	if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO = "00".
9-8	Reserved. Bits reserved for future use and should be set to zero.
7-0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4, 2, 0 Reserved.

The table below describes the dTD Buffer Page Pointer List.

Table 524. dTD Buffer Page Pointer List

Bit	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0,11-0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1,10-0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

64.6.6 Device operational model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus.

Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

64.6.6.1 Port state and control

After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0.

The following state diagram depicts the state of a USB 2.0 device.

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'.

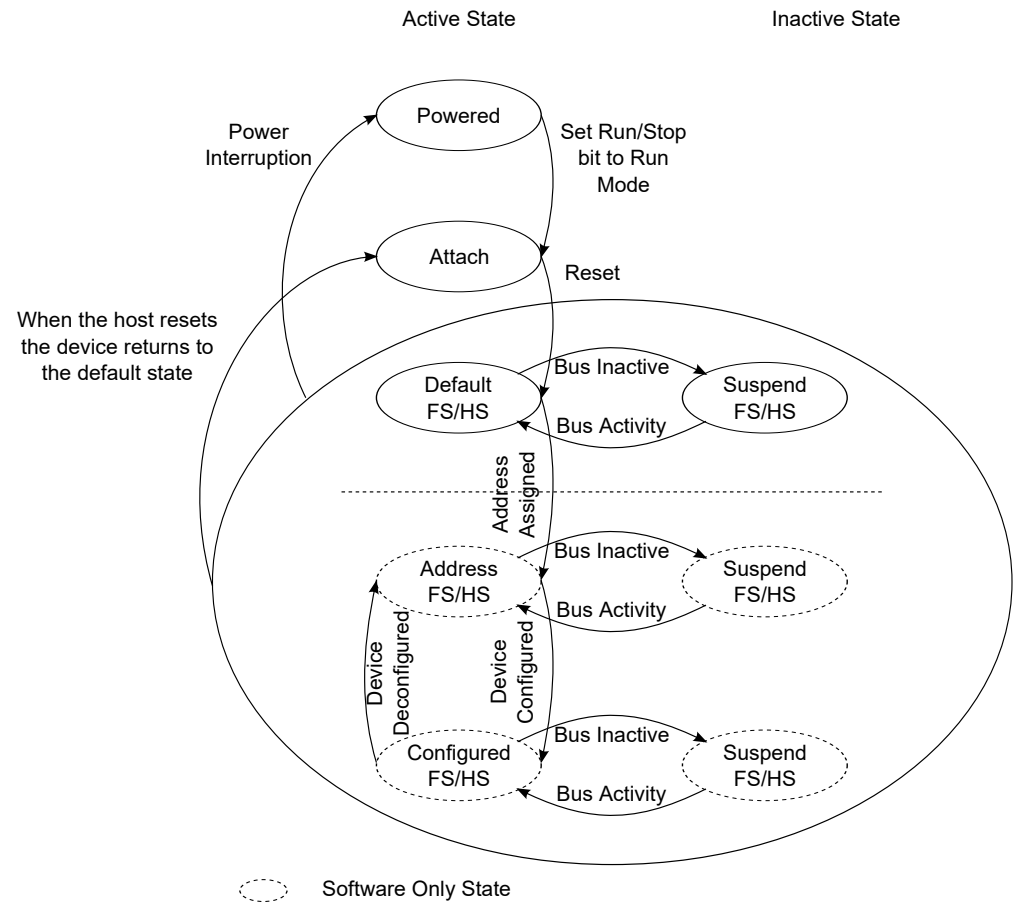


Figure 351. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

The following table describes the Device Controller State Information Bits.

Table 525. Device Controller State Information Bits

Bit	Register
DCSuspend	USBSTS
USB Reset Received	USBSTS
Port Change Detect	USBSTS
High-Speed Port	PORTSC1

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the USB_UOG_ENDPTCTRLx registers and initializing the associated queue heads.

64.6.6.1.1 Bus reset

A bus reset is used by the host to initialize downstream devices.

When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the USB_nENDPTSTAT register and writing the same value back to the USB_nENDPTSTAT register.

Clear all the endpoint complete status bits by reading the USB_nPTCOMPLETE register and writing the same value back to the USB_nENDPTCOMPLETE register.

Cancel all primed status by waiting until all bits in the USB_nENDPTPRIME are 0 and then writing 0xFFFFFFFF to USB_nENDPTFLUSH.

Read the reset bit in the USB_nPORTSC1 register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the USB_nPORTSC1 to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

64.6.6.1.2 Suspend and resume

Suspend state: operational model	Resume state: operational model
To conserve power, USB devices automatically enter the Suspended state when the device observes no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. The attached devices must be prepared to suspend at any time they are powered, regardless of whether they are assigned a nondefault address, are configured, or neither of the two	If the device controller is suspended, its operation is resumed when an active signal is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port by writing 1 to PORTSC1[FPR] while the device is in the Suspend state. Sending resume signal to an upstream port must cause the host to issue resume signaling and bring the suspended bus segment (one more device) back to the active condition.

Table continues on the next page...

Suspend state: operational model	Resume state: operational model
conditions exist. Bus activity may cease because of the host entering a Suspend mode of its own. In addition, a USB device must enter the Suspended state when the hub port it is attached to is disabled. A USB device exits Suspend mode when there is bus activity. It may also request the host to exit Suspend mode or selective suspend by using electrical signaling to indicate remote wake-up. The ability of a device to signal remote wake-up is optional. If the USB device is capable of remote wake-up signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wake-up signaling must be disabled. The device controller moves into the Suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the Suspend state, an interrupt notifies the DCD (assuming that PORTSC1[SUSP] is 1, which means, the device controller is suspended). The DCD response, when the device controller is suspended, is application-specific and may involve switching to a low-power operation. See the USB 2.0 specification for information about bus power limits in Suspend state. **NOTE** Review system level clocking issues defined in Clocking section for the clocking requirements of a suspended device controller.	**NOTE** The host must enable resume signaling, by using the set feature command defined in the device framework (chapter 9 of the USB 2.0 specification), before using it.

64.6.6.2 Managing endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device.

The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and *isochronous*. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The USB OTG device controller hardware supports up to 8 endpoint numbers.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. To support the 8 endpoint numbers, 16 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

64.6.6.2.1 Endpoint initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the USB_UOG_ENDPTCTRLx register.

Each 32-bit USB_UOG_ENDPTCTRLx is split into an upper and lower half. The lower half of USB_UOG_ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the USB_UOG_ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization. The following table shows the fields and values for the Device Controller Endpoint initialization.

Table 526. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 Control 01 Isochronous 10 Bulk 11 Interrupt
Endpoint Stall	0

64.6.6.2.2 Stalling

There are two occasions where the device controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the USB_UOG_ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the USB_UOG_ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the USB_UOG_ENDPTCTRLx register during operational mode must preserve the endpoint type field (that is, perform a read-modify-write).

The following table shows the response matrix for the Device Controller Stall.

Table 527. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'1'	None.	STALL

Table continues on the next page...

Table 527. Device Controller Stall Response Matrix (continued)

IN/OUT/PING packet received by a non-control endpoint.	'0'	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	'1'	None	STALL
IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

64.6.6.2.3 Data toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe.

For more information on data toggle, refer to the USB 2.0 specification.

64.6.6.2.3.1 Data toggle reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the USB_UOG_ENDPTCTRLx register.

This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

64.6.6.2.3.2 Data toggle inhibit

NOTE

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

64.6.6.2.3.3 Priming transmit endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH).

After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the USB_UOG_ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Because only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

64.6.6.2.3.4 Priming receive endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

64.6.6.3 Operational model for packet transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification.

At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "priming" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term "flushing" is used to describe the action of clearing a packet that was queued for execution.

64.6.6.3.1 Interrupt/Bulk endpoint operational model

The behaviors of the device controller for interrupt and bulk endpoints are identical.

All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 528. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3

Table continues on the next page...

Table 528. Variable Length Transfer Protocol Example (ZLT = 0) (continued)

511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

Table 529. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

NOTE

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

64.6.6.3.1.1 Interrupt/Bulk endpoint bus response matrix

The table below shows the response matrix for Interrupt/Bulk Endpoint Bus.

Table 530. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow

Table continues on the next page...

Table 530. Interrupt/Bulk Endpoint Bus Response Matrix (continued)

Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

NOTE

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

64.6.6.3.2 Control endpoint operation model

This section details the setup phase, data phase, status phase, and the control endpoint bus response matrix.

64.6.6.3.2.1 Setup phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

The setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

- Disable Setup Lockout by writing 1 to Setup Lockout Mode (SLOM) in USB_nUSBMODE. (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the Setup Lockout Mode As 0 will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting UBS_nENDSETUPSTAT to determine that a setup packet was received on a particular pipe:
  1. Write 1 to clear corresponding bit UBS_nENDSETUPSTAT.
  2. Write 1 to Setup Tripwire (SUTW) in USB_nUSBCMD register.
  3. Duplicate contents of dQH.SetupBuffer into local software byte array.
  4. Read Setup TripWire (SUTW) in USB_nUSBCMD register. (if set - continue; if cleared - go to 2)
  5. Write 0 to clear Setup Tripwire (SUTW) in USB nUSBCMD register.
  6. Process setup packet using local software byte array copy and execute status/handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

64.6.6.3.2.2 Data phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the USB.ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the USB_nENDPTPRIME register is zero and the associated bit in the USB_nENDPTSTAT register is a one. If a prime fails, i.e. The USB_nENDPTPRIME bit goes to zero and the USB_nENDPTSTAT bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (USB_nENDPTSTAT) to enforce data coherency with the setup packet.

NOTE

- The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

64.6.6.3.2.3 Status phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase.

The DCD must also perform the same checks of the USB.ENDPTSETUPSTAT as described above in the data phase.

NOTE

- The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

64.6.6.3.2.4 Control endpoint bus response matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

The table below shows the response matrix for the Control Endpoint Bus.

Table 531. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A

Table continues on the next page...

Table 531. Control Endpoint Bus Response Matrix (continued)

Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

64.6.6.3.3 Isochronous endpoint operational model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes.

Real time delivery by the device controller will be accomplished by the following:

- Exactly MULT Packets per (micro) Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro) frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - MULT counter reaches zero.
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - MULT counter reaches zero.
  - Non-MDATA Data PID is received**
    - ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
  - Overflow Error:
    - Packet received is > maximum packet length. [*Buffer Error* bit is set]
    - Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT
  - CRC Error [*Transaction Error* bit is set]

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

64.6.6.3.3.1 Isochronous pipe synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro) frame number (USB_UOG_FRINDEX register) can be used as a marker.

To cause a packet transfer to occur at a specific (micro) frame number [N], the DCD should interrupt on SOF during frame N-1. When the USB_UOG_FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro) frame N-1 so that the device controller will execute delivery during (micro) frame N.

NOTE

Priming an endpoint towards the end of (micro) frame N-1 will not guarantee delivery in (micro) frame N. The delivery may actually occur in (micro) frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

64.6.6.3.3.2 Isochronous endpoint bus response matrix

The following table shows the response matrix for the Isochronous Endpoint Bus.

Table 532. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. BS Error = Force Bit Stuff Error

   NULL Packet = Zero Length Packet

64.6.6.4 Managing queue heads

The following figure shows the End Point Queue Head.

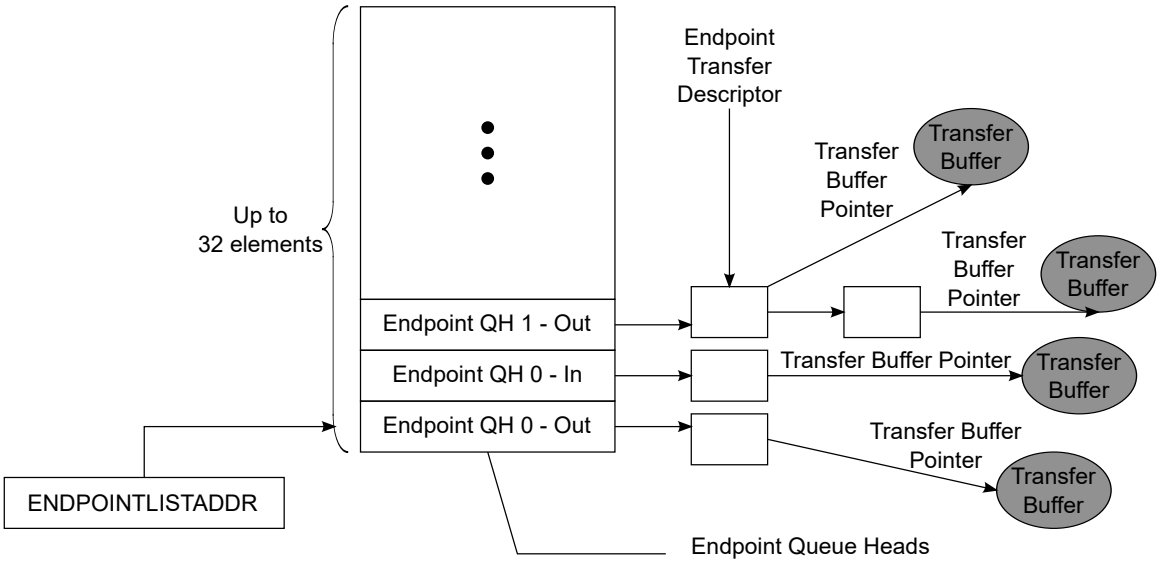


Figure 352. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by USB.ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 352. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors because pointers will no longer exist within the queue head once the dTD is retired (see section Software link pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational model for packet transfers, the dQH also contains the following parameters for the associated endpoint: Multiplier, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

64.6.6.4.1 Queue head initialization

One device queue head must be initialized for each active endpoint.

To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth and in conjunction with the USB Chapter 9 protocol.

NOTE

In FS mode, the multiplier field can only be 1 for ISO endpoints.

- Write the next dTD Terminate bit field to 1.
- Write the Active bit in the status field to 0.
- Write the Halt bit in the status field to 0.

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

64.6.6.4.2 Operational model for setup transfers

As discussed in section Control endpoint operation model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

- The acknowledge must occur before continuing to process the setup packet.
- After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

64.6.6.5 Managing transfers with transfer descriptors

These sections detail how to build, execute, and complete transfers with transfer descriptors.

64.6.6.5.1 Software link pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head.

This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list. The following figure shows the Software Link Pointers.

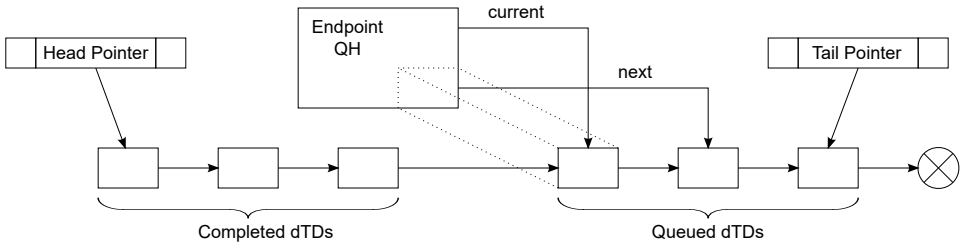


Figure 353. Software Link Pointers

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers, but it still remains the responsibility of the DCD to maintain the pointers.

64.6.6.5.2 Building a transfer descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer.

Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to "00000"

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to 1.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to 1 and all remaining status bits set to 0.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

64.6.6.5.3 Executing a transfer descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty: Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding).

- Case 1: Link list is empty
    1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
    2. Clear active & halt bit in dQH (in case set from a previous error).
    3. Prime endpoint by writing 1 to correct bit position in ENDPTPRIME. Software should check that the dTD/dQH content in external memory is updated before setting the bit in ENDPTPRIME register.
- Case 2: Link list is not empty
    1. Add dTD to end of linked list.
    2. Read correct prime bit in ENDPTPRIME - if 1 DONE.
    3. Set ATDTW bit in USBCMD register to 1.
    4. Read correct status bit in ENDPTSTAT. (store in tmp. variable for later)
    5. Read ATDTW bit in USBCMD register.
        - If 0 go to 3.
        - If 1 continue to 6.
    6. Write ATDTW bit in USBCMD register to 0.
    7. If status bit read in (3) is 1 DONE.
    8. If status bit read in (3) is 0 then go to Case 1: Step 1.

64.6.6.5.4 Transfer completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device error matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

64.6.6.5.5 Flushing/De-priming an endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer.

There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are '0'.
   - Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
3. Read ENDPTSTAT register to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
   - Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH register. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

64.6.6.5.6 Device error matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 533. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated. The table below describes the errors.

Table 534. Error Descriptions

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length. ** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the "dead" (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

64.7 USB non-core memory map/register definition

64.7.1 USBNC register descriptions

There are two kinds of registers in the USB module: The USB core registers and USB non-core registers. The USB core registers are used to control USB core functions and are more independent of USB features. The USB controller core has its own core registers. The USB non-core registers are additional to USB core registers and are more dependent on the USB features. The USB non-core registers support the 32,16, and 8-bit register access.

This section describes only the USB non-core registers. For detailed descriptions of the USB core registers, see Register interface.

NOTE

For reserved bits, preserve the value when writing (read its reset value, then write this value back).

64.7.1.1 USBNC memory map

USBHS1.USBNC base address: 4010_B200h

Offset	Register	Width (In bits)	Access	Reset value
0h	USB OTG Control 1 (CTRL1)	32	RW	3000_1000h
4h	USB OTG Control 2 (CTRL2)	32	RW	5F00_0000h
10h	USB Host HSIC Control (HSIC_CTRL)	32	RW	1000_4084h

64.7.1.2 USB OTG Control 1 (CTRL1)

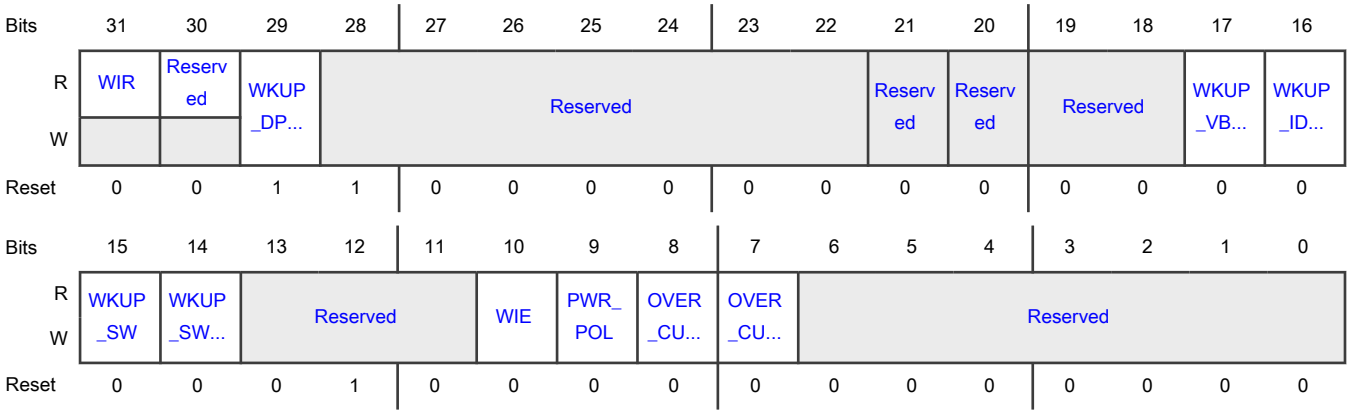
Offset

Register	Offset
CTRL1	0h

Function

Controls the integration specific features of the USB OTG1 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Diagram



Fields

Field	Function
31 WIR	Wake-up Interrupt Request Indicates that a wake-up interrupt request is received on the OTG1 port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 0b - No request received 1b - Request received
30 —	Reserved
29 WKUP_DPDM_EN	Wake-up on DPDM Change Enable 0b - DPDM changes wake-up to be disabled only when VBUS is 0 1b - DPDM changes wake-up to be enabled, it is for device only
28-22 —	Reserved
21 —	Reserved
20 —	Reserved
19-18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 WKUP_VBUS_EN	Wake-up on VBUS Change Enable 0b - Disables 1b - Enables
16 WKUP_ID_EN	Wake-up on ID Change Enable 0b - Disables 1b - Enables
15 WKUP_SW	Software Wake-up 0b - Inactive 1b - Force wake-up
14 WKUP_SW_EN	Software Wake-up Enable 0b - Disables 1b - Enables
13-11 —	Reserved
10 WIE	Wake-up Interrupt Enable Enables or disables the OTG1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode. 0b - Interrupt Disabled 1b - Interrupt Enabled
9 PWR_POL	Power Polarity Should be set according to PMIC Power Pin polarity. 0b - PMIC Power Pin is Low active. 1b - PMIC Power Pin is High active.
8 OVER_CUR_POL	Polarity of Overcurrent Configures the polarity of OTG1 port overcurrent event. 0b - High active (high on this signal represents an overcurrent condition) 1b - Low active (low on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable Overcurrent Detection 0b - Enables 1b - Disables
6-0	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	

64.7.1.3 USB OTG Control 2 (CTRL2)

Offset

Register	Offset
CTRL2	4h

Function

Controls the integration specific features of the USB module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wakeup functionality.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UTMI_CL...	Reserved			Reserved								Reserved			
W	W1C															
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												LOWSPEE...	AUTURES...	VBUS_SOURCE_SEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 UTMI_CLK_VLD	UTMI Clock Valid Indicates whether the UTMI clock to the USB PHY is valid. Write 1 to clear 0b - Default
30-28 —	Reserved
27-20 —	Reserved 0001_1110b - Default

Table continues on the next page...

Table continued from the previous page...

Field	Function
19-4 —	Reserved
3 LOWSPEED_EN	Low Speed Enable Set if AUTURESUME_EN is set and works on low speed. 0b - Default
2 AUTURESUME_EN	Auto Resume Enable It is for UTMI PHY host mode only (UH core has no this feature since HSIC PHY does not support auto resume). To set USB, it will send resume with 32 kHz clock when it detects remote wakeup from device. This feature is useful if device drive a very short remote wakeup (minimal value is 1 ms for USB2 spec) and system 24 MHz is off during suspend mode. 0b - Default
1-0 VBUS_SOURCE_SEL	VBUS Source Select Selects VBUS source When detecting VBUS wakeup event, it is for UTMI PHY only (UH core has no such feature). 00b - vbus_valid 01b - sess_valid 10b - sess_valid 11b - sess_valid

64.7.1.4 USB Host HSIC Control (HSIC_CTRL)

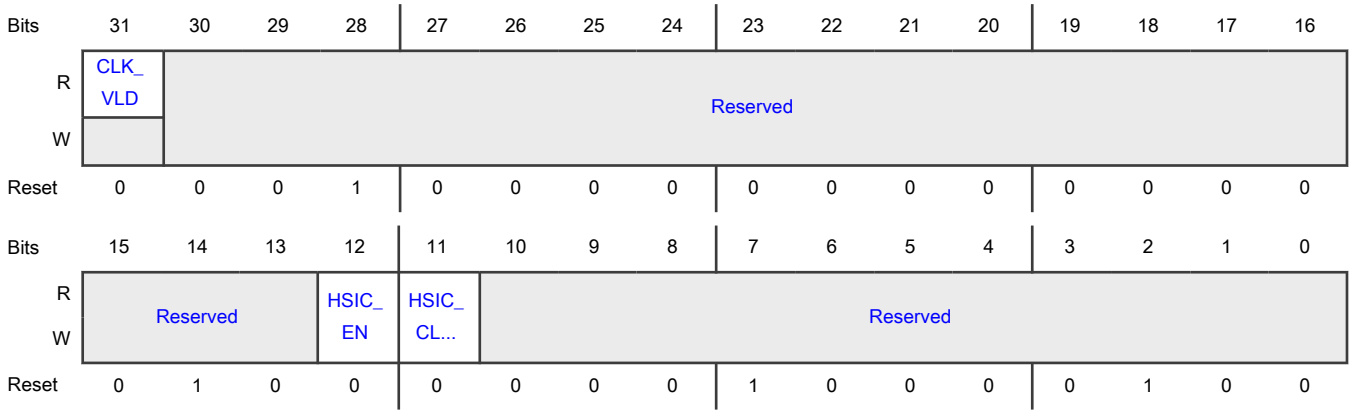
Offset

Register	Offset
HSIC_CTRL	10h

Function

Controls Host high speed IC configuration. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control.

Diagram



Fields

Field	Function
31 CLK_VLD	Clock Valid Indicates whether Host HSIC clock is valid. 0b - Invalid 1b - Valid
30-13 —	Reserved
12 HSIC_EN	Host HSIC Enable 0b - Disabled 1b - Enabled
11 HSIC_CLK_ON	HSIC Clock ON Force Host HSIC module 480M clock on, even when in Host is in suspend mode. 0b - Inactive 1b - Active
10-0 —	Reserved

64.8 USB core memory map/register definition

64.8.1 USB register descriptions

64.8.1.1 USB memory map

USBHS1.USBC base address: 4010_B000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Identification (ID)	32	R	E4A1_FA05h
4h	Hardware General (HWGENERAL)	32	R	0000_0015h
8h	Host Hardware Parameters (HWHOST)	32	R	1002_0001h
Ch	Device Hardware Parameters (HWDEVICE)	32	R	0000_0011h
10h	TX Buffer Hardware Parameters (HWTXBUF)	32	R	8005_0808h
14h	RX Buffer Hardware Parameters (HWRXBUF)	32	R	0000_0808h
80h	General Purpose Timer #0 Load (GPTIMER0LD)	32	RW	0000_0000h
84h	General Purpose Timer #0 Controller (GPTIMER0CTRL)	32	RW	0000_0000h
88h	General Purpose Timer #1 Load (GPTIMER1LD)	32	RW	0000_0000h
8Ch	General Purpose Timer #1 Controller (GPTIMER1CTRL)	32	RW	0000_0000h
90h	System Bus Config (SBUSCFG)	32	RW	0000_0002h
100h	Capability Registers Length (CAPLENGTH)	8	R	40h
102h	Host Controller Interface Version (HCIVERSION)	16	R	0100h
104h	Host Controller Structural Parameters (HCSPARAMS)	32	R	0001_0011h
108h	Host Controller Capability Parameters (HCCPARAMS)	32	R	0000_0006h
120h	Device Controller Interface Version (DCIVERSION)	16	R	0001h
124h	Device Controller Capability Parameters (DCCPARAMS)	32	R	0000_0188h
140h	USB Command (USBCMD)	32	RW	0008_0000h
144h	USB Status (USBSTS)	32	RW	See section
148h	Interrupt Enable (USBINTR)	32	RW	0000_0000h
14Ch	USB Frame Index (FRINDEX)	32	RW	0000_0000h
154h	Device Address (DEVICEADDR)	32	RW	0000_0000h
154h	Frame List Base Address (PERIODICLISTBASE)	32	RW	0000_0000h
158h	Next Asynch. Address (ASYNCLISTADDR)	32	RW	0000_0000h
158h	Endpoint List Address (ENDPTLISTADDR)	32	RW	0000_0000h
160h	Programmable Burst Size (BURSTSIZE)	32	RW	0000_0808h
164h	TX FIFO Fill Tuning (TXFILLTUNING)	32	RW	0000_0000h
178h	Endpoint NAK (ENDPTNAK)	32	RW	0000_0000h
17Ch	Endpoint NAK Enable (ENDPTNAKEN)	32	RW	0000_0000h
180h	Configure Flag (CONFIGFLAG)	32	R	0000_0001h
184h	Port Status & Control (PORTSC1)	32	RW	1C00_0004h
1A4h	On-The-Go Status & Control (OTGSC)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
1A8h	USB Device Mode (USBMODE)	32	RW	0000_5000h
1ACh	Endpoint Setup Status (ENDPTSETUPSTAT)	32	RW	0000_0000h
1B0h	Endpoint Prime (ENDPTPRIME)	32	RW	0000_0000h
1B4h	Endpoint Flush (ENDPTFLUSH)	32	RW	0000_0000h
1B8h	Endpoint Status (ENDPTSTAT)	32	R	0000_0000h
1BCh	Endpoint Complete (ENDPTCOMPLETE)	32	RW	0000_0000h
1C0h	Endpoint Control 0 (ENDPTCTRL0)	32	RW	0080_0080h
1C4h	Endpoint Control 1 (ENDPTCTRL1)	32	RW	0000_0000h
1C8h	Endpoint Control 2 (ENDPTCTRL2)	32	RW	0000_0000h
1CCh	Endpoint Control 3 (ENDPTCTRL3)	32	RW	0000_0000h
1D0h	Endpoint Control 4 (ENDPTCTRL4)	32	RW	0000_0000h
1D4h	Endpoint Control 5 (ENDPTCTRL5)	32	RW	0000_0000h
1D8h	Endpoint Control 6 (ENDPTCTRL6)	32	RW	0000_0000h
1DCh	Endpoint Control 7 (ENDPTCTRL7)	32	RW	0000_0000h

64.8.1.2 Identification (ID)

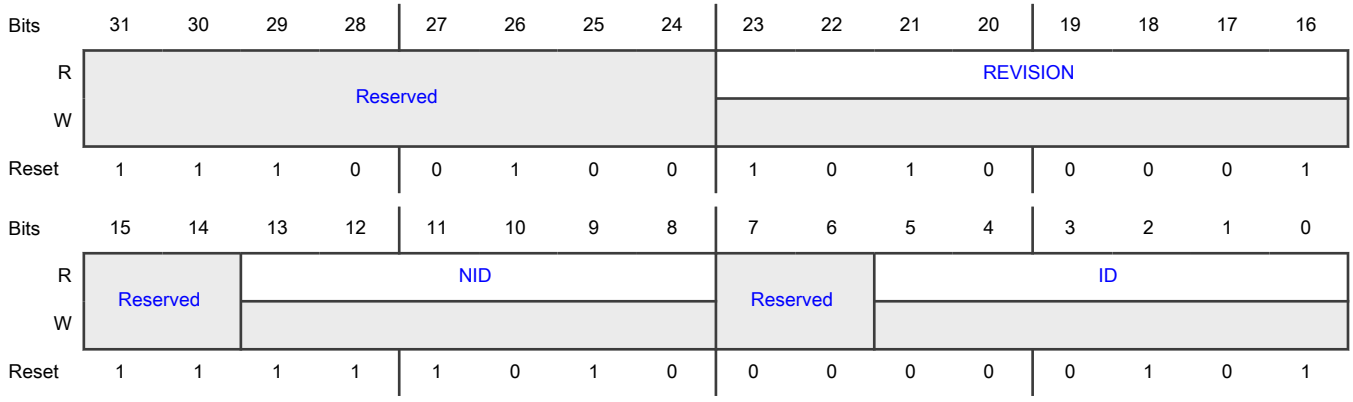
Offset

Register	Offset
ID	0h

Function

Identifies the USB 2.0 High-Speed core and its revision.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 REVISION	Revision Number of the Controller Core
15-14 —	Reserved
13-8 NID	Complement Version of ID
7-6 —	Reserved
5-0 ID	Configuration Number This number is set to 0x05 and indicates that the peripheral is USB 2.0 High-Speed core.

64.8.1.3 Hardware General (HWGENERAL)

Offset

Register	Offset
HWGENERAL	4h

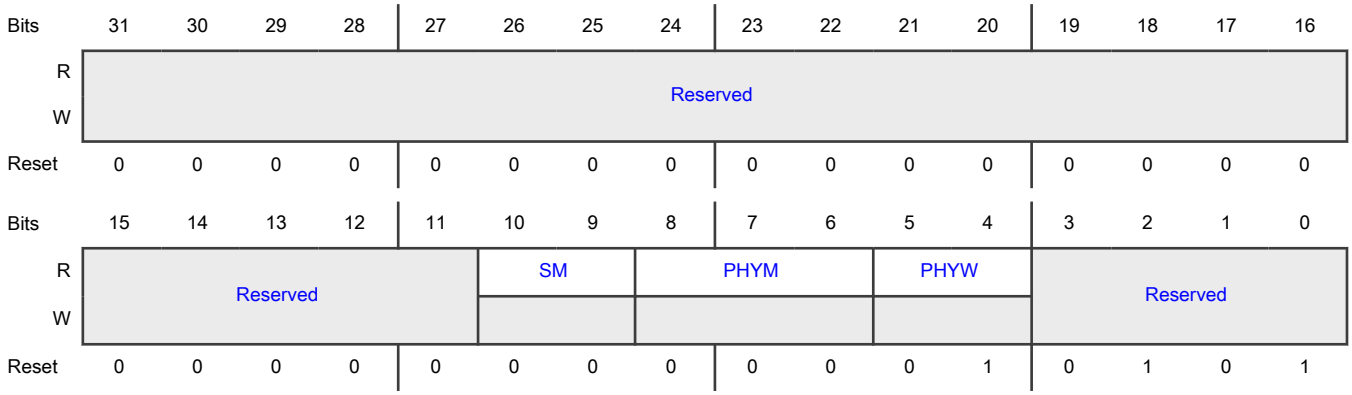
Function

General hardware parameters as defined in System Level Issues and Core Configuration.

NOTE

The reset value could vary from instance to instance. See the detail in bit field description and ignore reset value in summary table in this case!

Diagram



Fields

Field	Function
31-11 —	Reserved
10-9 SM	Serial interface mode capability 00b - No Serial Engine, always use parallel signalling 01b - Serial Engine present, always use serial signalling for FS/LS 10b - Software programmable - Reset to use parallel signalling for FS/LS 11b - Software programmable - Reset to use serial signalling for FS/LS
8-6 PHYM	Transceiver Type 000b - UTMI/UMTI+ 001b - ULPI DDR 010b - ULPI 011b - Serial Only 100b - Software programmable - reset to UTMI/UTMI+ 101b - Software programmable - reset to ULPI DDR 110b - Software programmable - reset to ULPI 111b - Software programmable - reset to Serial
5-4 PHYW	Data width of the transceiver connected to the controller core 00b - 8 bit wide data bus (Software non-programmable) 01b - 16 bit wide data bus (Software non-programmable) 10b - Reset to 8 bit wide data bus (Software programmable) 11b - Reset to 16 bit wide data bus (Software programmable)
3-0 —	Reserved

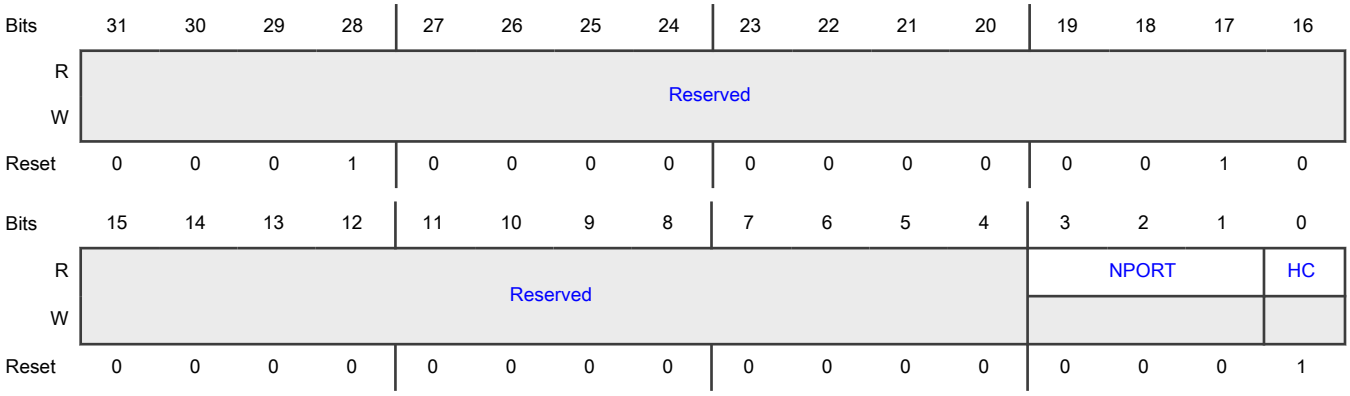
64.8.1.4 Host Hardware Parameters (HWHOST)

Offset

Register	Offset
HWHOST	8h

Function
Indicates host hardware paramters such as host operation mode support and number of downstream ports.

Diagram



Fields

Field	Function
31-4 —	Reserved
3-1 NPORT	The Number of downstream ports supported by the host controller is (NPORT+1) **NOTE** When these bits value is '000', it indicates a single-port host controller.
0 HC	Host Capable Indicating whether host operation mode is supported or not. 0b - Not supported 1b - Supported

64.8.1.5 Device Hardware Parameters (HWDEVICE)

Offset

Register	Offset
HWDEVICE	Ch

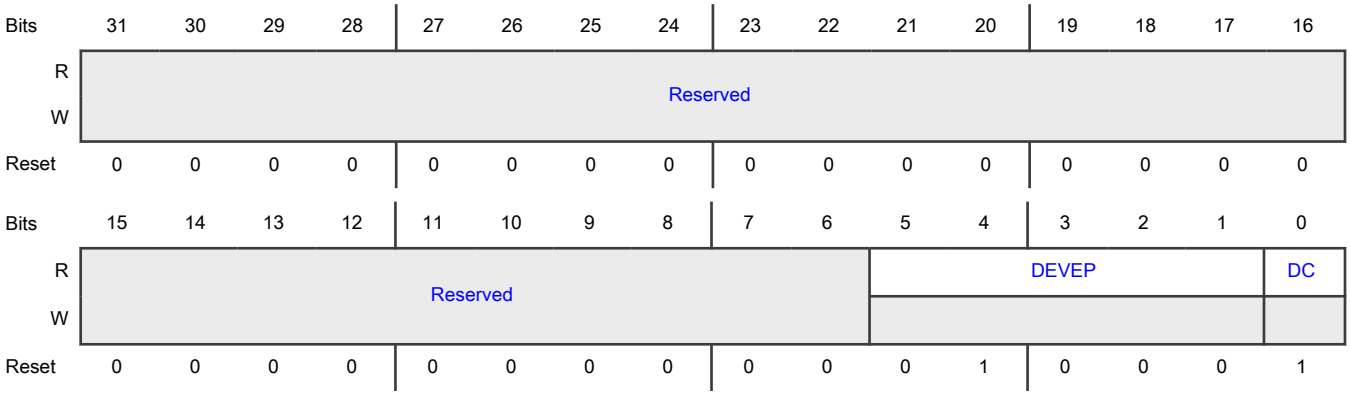
Function

Indicates device hardware paramters such as device operation mode support and number of device endpoints.

NOTE

This register is only available in OTG core.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-1 DEVEP	Device Endpoint Number
0 DC	Device Capable Indicating whether device operation mode is supported or not. 0b - Not supported 1b - Supported

64.8.1.6 TX Buffer Hardware Parameters (HWTXBUF)

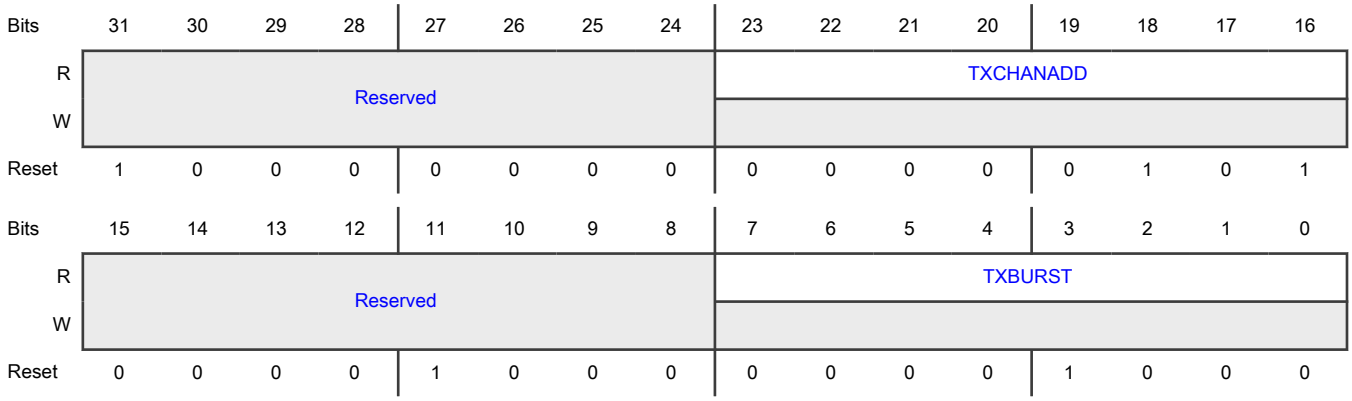
Offset

Register	Offset
HWTXBUF	10h

Function

Indicates TX buffer hardware parameters such as burst size, FIFO buffer size, and so on.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 TXCHANADD	TX FIFO Buffer size is: (2^TXCHANADD) * 4 Bytes These bits are set to '08h', so buffer size is 256*4 Bytes. For the OTG controller operating in device mode, this is the FIFO buffer size per endpoint. As the OTG controller has 8 TX endpoint, there are 8 of these buffers. For the OTG controller operating in host mode, or for Host-only controller, the entire buffer memory is used as a single TX buffer. Therefore, there is only 1 of this buffer.
15-8 —	Reserved
7-0 TXBURST	Default burst size for memory to TX buffer transfer This is reset value of TXPBURST bits in USB core register USB_n_BURSTSIZE. See BURSTSIZE.

64.8.1.7 RX Buffer Hardware Parameters (HWRXBUF)

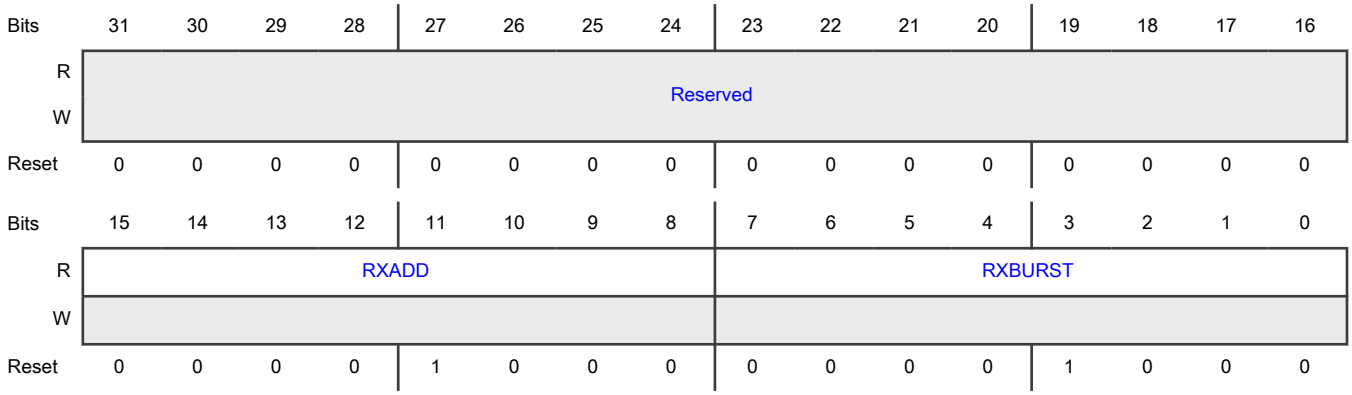
Offset

Register	Offset
HWRXBUF	14h

Function

Indicates RX buffer hardware parameters such as burst size, FIFO buffer size, and so on.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 RXADD	Buffer total size for all receive endpoints is (2^RXADD) RX Buffer size is: (2^RXADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. There is a single Receive FIFO buffer in the USB controller. The buffer is shared for all endpoints for the OTG controller in device mode.
7-0 RXBURST	Default burst size for memory to RX buffer transfer This is reset value of RXPBURST bits in USB core register USB_n_BURSTSIZE. See BURSTSIZE.

64.8.1.8 General Purpose Timer #0 Load (GPTIMER0LD)

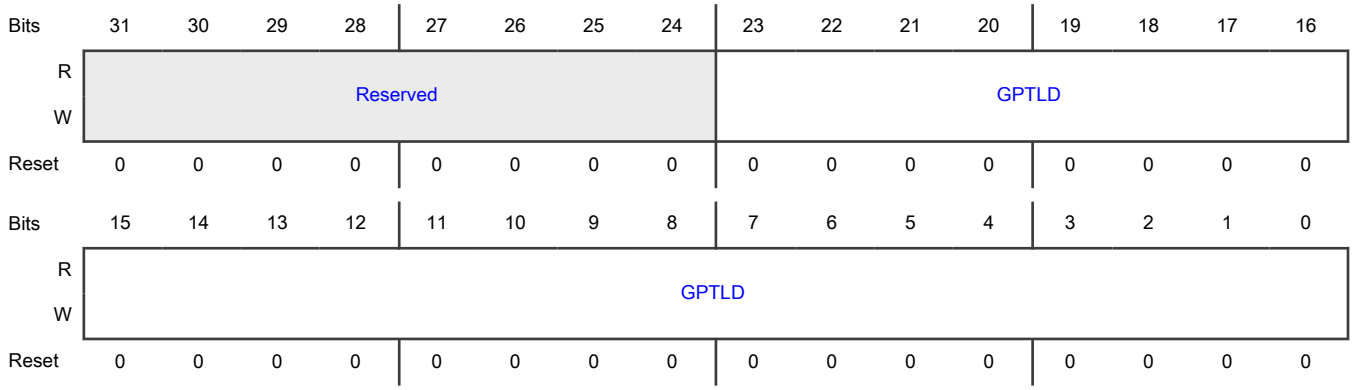
Offset

Register	Offset
GPTIMER0LD	80h

Function

Controls load value of the count timer in register n_GPTIMER0CTRL. See GPTIMER0CTRL.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE** Max value is 0xFFFFFF or 16.777215 seconds.

64.8.1.9 General Purpose Timer #0 Controller (GPTIMER0CTRL)

Offset

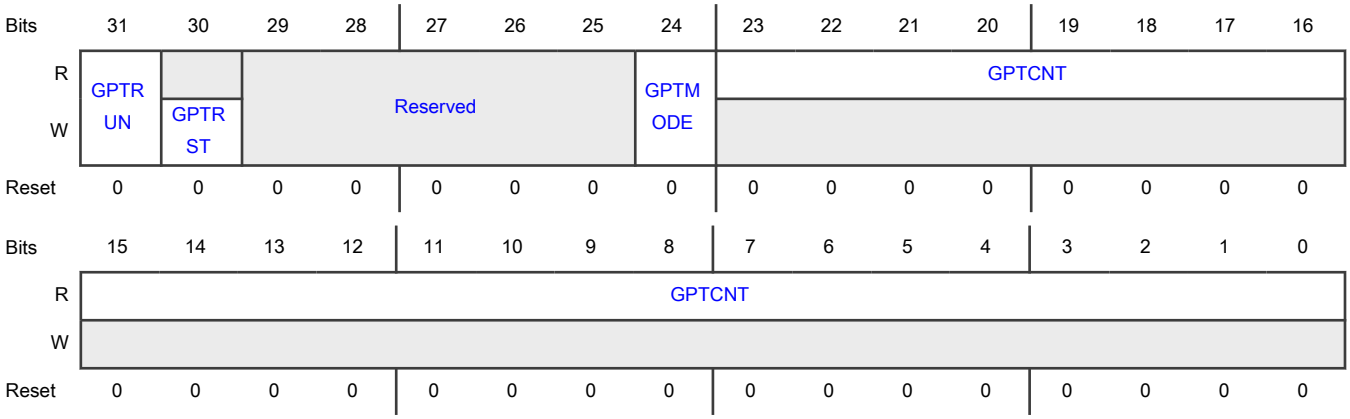
Register	Offset
GPTIMER0CTRL	84h

Function

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI0 bit in n_USBSTS register (See USBSTS), interrupt enable bit is TIE0 bit in n_USBINTR register. (See USBINTR .)

Diagram



Fields

Field	Function
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0b - Stop counting 1b - Run
30 GPTRST	General Purpose Timer Reset 0b - No action 1b - Load counter value from GPTLD bits in n_GPTIMER0LD
29-25 —	Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software; In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0b - One Shot Mode 1b - Repeat Mode
23-0 GPTCNT	General Purpose Timer Counter This field is the count value of the countdown timer.

64.8.1.10 General Purpose Timer #1 Load (GPTIMER1LD)

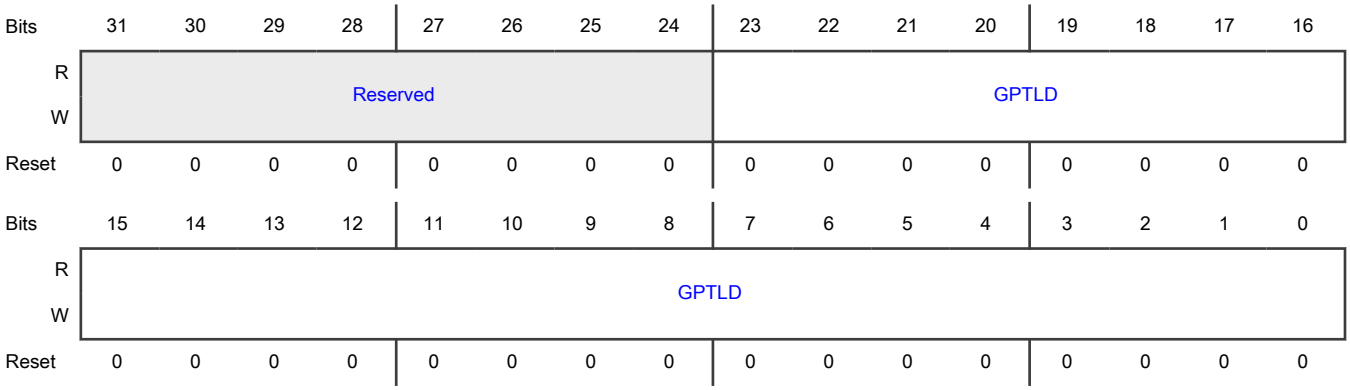
Offset

Register	Offset
GPTIMER1LD	88h

Function

This register controls load value of the count timer in register n_GPTIMER1CTRL. See GPTIMER1CTRL .

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE** Max value is 0xFFFFFF or 16.777215 seconds.

64.8.1.11 General Purpose Timer #1 Controller (GPTIMER1CTRL)

Offset

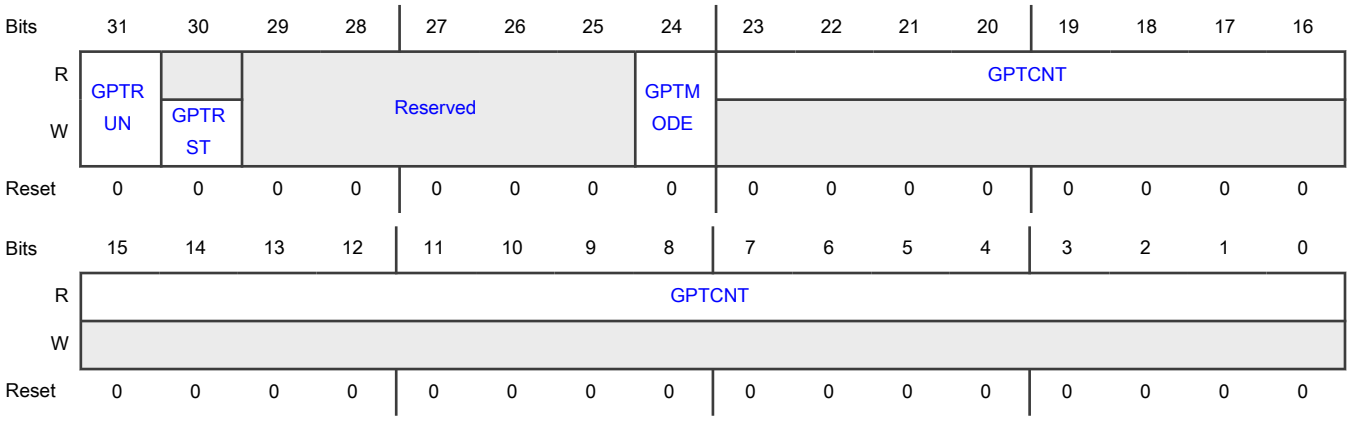
Register	Offset
GPTIMER1CTRL	8Ch

Function

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI1 bit in USB_n_USBSTS register (See USBSTS), interrupt enable bit is TIE1 bit in n_USBINTR register (See USBINTR).

Diagram



Fields

Field	Function
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0b - Stop counting 1b - Run
30 GPTRST	General Purpose Timer Reset 0b - No action 1b - Load counter value from GPTLD bits in USB_n_GPTIMER0LD
29-25 —	Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0b - One Shot Mode 1b - Repeat Mode
23-0 GPTCNT	General Purpose Timer Counter This field is the count value of the countdown timer.

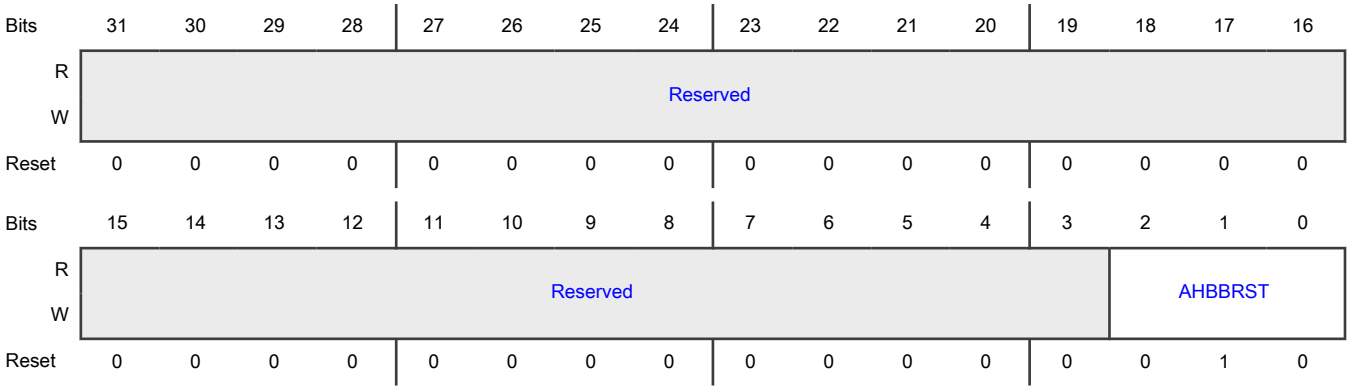
64.8.1.12 System Bus Config (SBUSCFG)

Offset

Register	Offset
SBUSCFG	90h

Function
This register defines the system bus configuration.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 AHBBRST	AHB master interface Burst configuration These bits control AHB master transfer type sequence (or priority). **NOTE** This register overrides n_BURSTSIZE register when its value is not zero. 000b - Incremental burst of unspecified length only 001b - INCR4 burst, then single transfer 010b - INCR8 burst, INCR4 burst, then single transfer 011b - INCR16 burst, INCR8 burst, INCR4 burst, then single transfer 100b - Reserved, don't use 101b - INCR4 burst, then incremental burst of unspecified length 110b - INCR8 burst, INCR4 burst, then incremental burst of unspecified length 111b - INCR16 burst, INCR8 burst, INCR4 burst, then incremental burst of unspecified length

64.8.1.13 Capability Registers Length (CAPLENGTH)

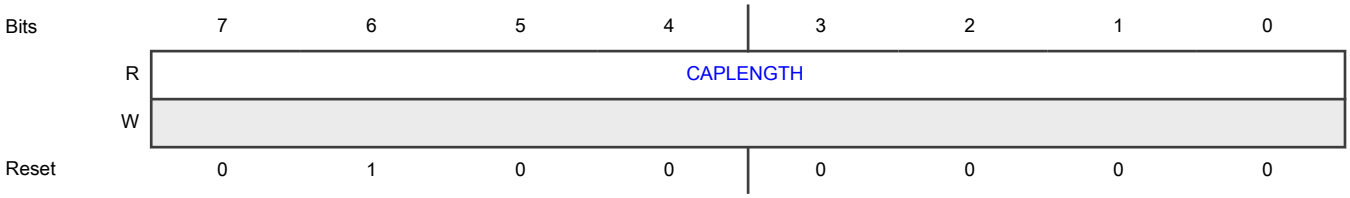
Offset

Register	Offset
CAPLENGTH	100h

Function

The Capability Registers Length register contains the address offset to the Operational registers relative to the CAPLENGTH register.

Diagram



Fields

Field	Function
7-0 CAPLENGTH	These bits are used as an offset to add to register base to find the beginning of the Operational Register. Default value is '40h'.

64.8.1.14 Host Controller Interface Version (HCIVERSION)

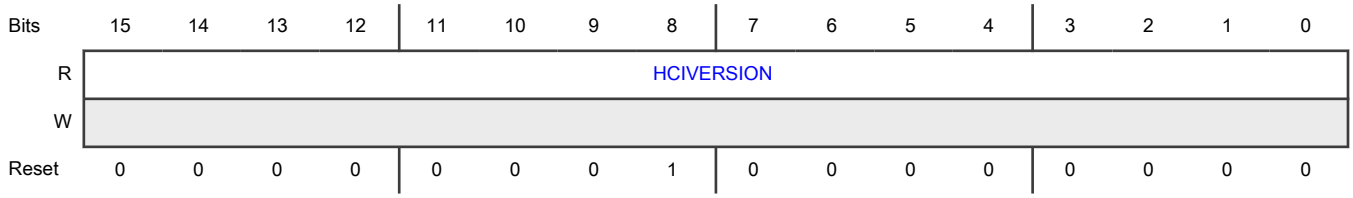
Offset

Register	Offset
HCIVERSION	102h

Function

This is a 2-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.

Diagram



Fields

Field	Function
15-0 HCIVERSION	Host Controller Interface Version Number Default value is '10h', which means EHCI rev1.0.

64.8.1.15 Host Controller Structural Parameters (HCSPARAMS)

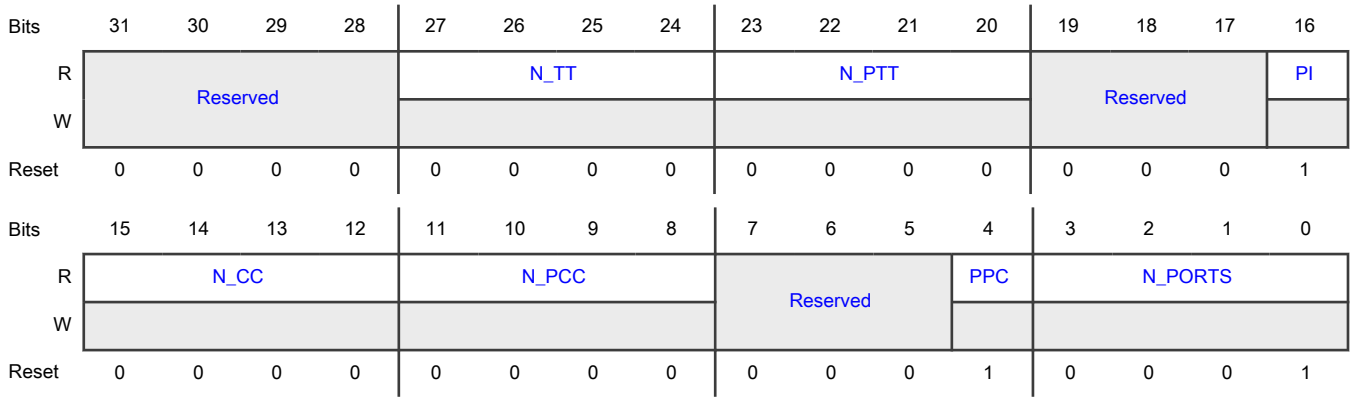
Offset

Register	Offset
HCSPARAMS	104h

Function

The following figure shows the port steering logic capabilities of Host Control Structural Parameters (n_HCSPARAMS).

Diagram



Fields

Field	Function
31-28 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
27-24 N_TT	Number of Transaction Translators This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. These bits would be set to '0001b' for Multi-Port Host, and '0000b' for Single-Port Host.
23-20 N_PTT	Number of Ports per Transaction Translator This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. These bits would be set to equal N_PORTS for Multi-Port Host, and '0000b' for Single-Port Host.
19-17 —	Reserved
16 PI	Port Indicators (P INDICATOR) This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator. This bit is "1b" in all controller core.
15-12 N_CC	Number of Companion Controller This field indicates the number of companion controllers associated with this USB2.0 host controller. These bits are '0000b' in all controller core. 0000b - There is no internal Companion Controller and port-ownership hand-off is not supported 0001b - There are internal companion controller(s) and port-ownership hand-offs is supported
11-8 N_PCC	Number of Ports per Companion Controller This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. These bits are '0000b' in all controller core.
7-5 —	Reserved
4 PPC	Port Power Control This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 N_PORTS	Number of Downstream Ports This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. These bits are always set to '0001b' because all controller cores are Single-Port Host.

64.8.1.16 Host Controller Capability Parameters (HCCPARAMS)

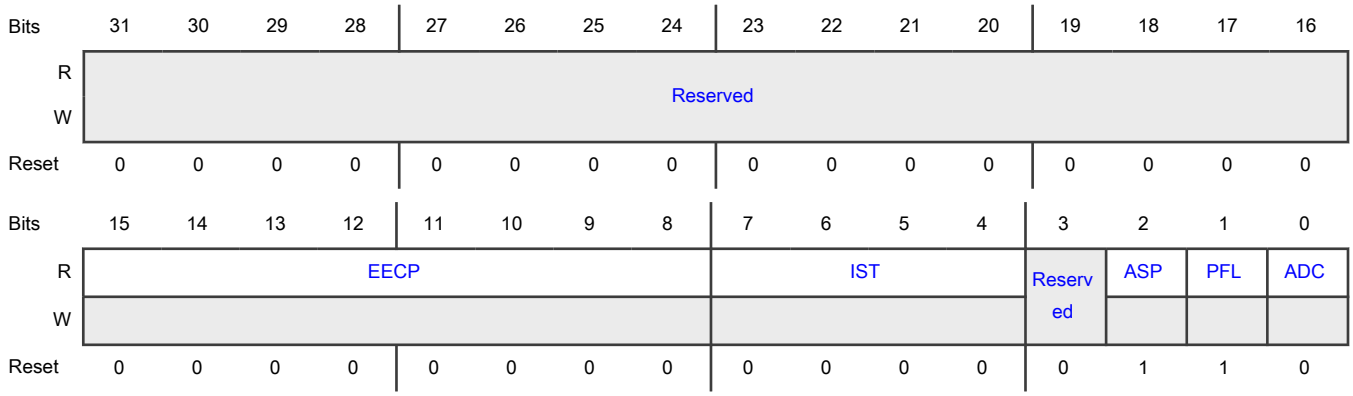
Offset

Register	Offset
HCCPARAMS	108h

Function

This register identifies multiple mode control (time-base bit functionality), addressing capability.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 EECP	EHCI Extended Capabilities Pointer This field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE These bits are set '00h' in all controller core.
7-4 IST	Isochronous Scheduling Threshold This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. These bits are set '00h' in all controller core.
3 —	Reserved
2 ASP	Asynchronous Schedule Park Capability If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. **NOTE** ASP bit reset value: '00b' for OTG controller core, '11b' for Host-only controller core.
1 PFL	Programmable Frame List Flag If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This bit is set '1b' in all controller core.
0 ADC	64-bit Addressing Capability This bit is set '0b' in all controller core, no 64-bit addressing capability is supported.

64.8.1.17 Device Controller Interface Version (DCIVERSION)

Offset

Register	Offset
DCIVERSION	120h

Function

This register indicates the two-byte BCD encoding of the device controller interface version number.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DCIVERSION															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Function
15-0 DCIVERSION	Device Controller Interface Version Number Default value is '01h', which means rev0.1.

64.8.1.18 Device Controller Capability Parameters (DCCPARAMS)

Offset

Register	Offset
DCCPARAMS	124h

Function

These fields describe the overall device capability of the controller.

NOTE

This register is only available in OTG controller core.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							HC	DC	Reserved			DEN			
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

Fields

Field	Function
31-9 —	Reserved
8 HC	Host Capable When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6-5 —	Reserved
4-0 DEN	Device Endpoint Number This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0 - 15.

64.8.1.19 USB Command (USBCMD)

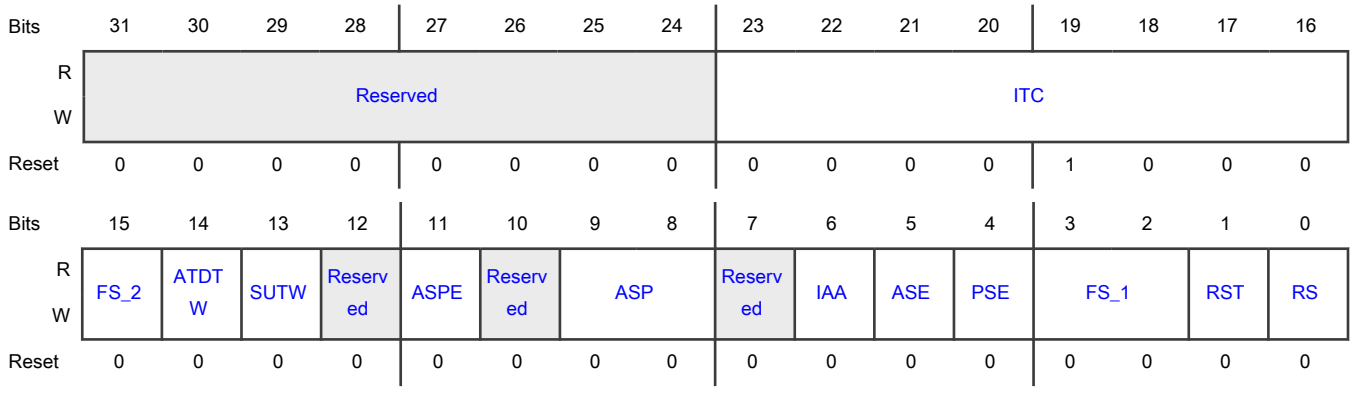
Offset

Register	Offset
USBCMD	140h

Function

The Command Register indicates the command to be executed by the serial bus host/device controller. Writing to the register causes a command to be executed.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 ITC	Interrupt Threshold Control The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 0000_0000b - Immediate (no threshold) 0000_0001b - 1 micro-frame 0000_0010b - 2 micro-frames 0000_0100b - 4 micro-frames 0000_1000b - 8 micro-frames 0001_0000b - 16 micro-frames 0010_0000b - 32 micro-frames 0100_0000b - 64 micro-frames
15 FS_2	Frame List Size [host mode only] This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. Specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. This field defines bit 2 of the following values. USBCMD[FS_1] defines bits 1:0. 000b - 1024 elements (4096 bytes) 001b - 512 elements (2048 bytes) 010b - 256 elements (1024 bytes) 011b - 128 elements (512 bytes) 100b - 64 elements (256 bytes) 101b - 32 elements (128 bytes) 110b - 16 elements (64 bytes) 111b - 8 elements (32 bytes)
14 ATDTW	Add dTD TripWire[device mode only] This bit is used as a semaphore to ensure proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit would also be cleared by hardware when state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 SUTW	Setup TripWire [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (SLOM bit in USB core register n_USBMODE, see USBMODE) then there is a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software. This bit would also be cleared by hardware when a hazard detected.
12 —	Reserved
11 ASPE	Asynchronous Schedule Park Mode Enable If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. **NOTE** ASPE bit reset value: '0b' for OTG controller .
10 —	Reserved
9-8 ASP	Asynchronous Schedule Park Mode Count If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is Read-Only. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in all controller core.
7 —	Reserved
6 IAA	Interrupt on Async Advance Doorbell This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable This bit controls whether the host controller skips processing the Asynchronous Schedule. Only the host controller uses this bit. 0b - Do not process the Asynchronous Schedule 1b - Use the ASYNCLISTADDR register to access the Asynchronous Schedule
4 PSE	Periodic Schedule Enable This bit controls whether the host controller skips processing the Periodic Schedule. Only the host controller uses this bit. 0b - Do not process the Periodic Schedule 1b - Use the PERIODICLISTBASE register to access the Periodic Schedule
3-2 FS_1	Frame List Size Specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. This field defines bits 1:0 of the following values. USBCMD[FS_2] defines bit 2. 000b - 1024 elements (4096 bytes) 001b - 512 elements (2048 bytes) 010b - 256 elements (1024 bytes) 011b - 128 elements (512 bytes) 100b - 64 elements (256 bytes) 101b - 32 elements (128 bytes) 110b - 16 elements (64 bytes) 111b - 8 elements (32 bytes)
1 RST	Controller Reset Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host operation mode: When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device operation mode:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, because the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop Host operation mode: When set to '1b', the Controller proceeds with the execution of the schedule. The Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device operation mode: Writing a one to this bit will cause the controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Writing a 0 to this will cause a detach event. 0b - Stop 1b - Run

64.8.1.20 USB Status (USBSTS)

Offset

Register	Offset
USBSTS	144h

Function

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						TI1	TI0	Reserved							NAKI
W							W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AS	PS	RCL	HCH	Reserved	ULPII	Reserved	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
W						W1C		W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	u	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25 TI1	General Purpose Timer Interrupt 1 (GPTINT1) This bit is set when the counter in the GPTIMER1CTRL register transitions to zero, writing a one to this bit will clear it.
24 TI0	General Purpose Timer Interrupt 0 (GPTINT0) This bit is set when the counter in the GPTIMER0CTRL register transitions to zero, writing a one to this bit clears it.
23-17 —	Reserved
16 NAKI	NAK Interrupt Bit This bit is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when all Enabled TX/RX Endpoint NAK bits are cleared.
15 AS	Asynchronous Schedule Status This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
14 PS	Periodic Schedule Status This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately

Table continues on the next page...

Table continued from the previous page...

Field	Function
	disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
13 RCL	Reclamation This is a read-only status bit used to detect an empty asynchronous schedule. Only used in the host operation mode.
12 HCH	HCHalted This bit is a zero whenever the Run/Stop bit is a one. The Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Controller hardware (for example, an internal error). Only used in the host operation mode. Default value is '0b' for OTG core . This is because OTG core is not operating as host in default. See CM bit in USB_n_USBMODE register. **NOTE** HCH bit reset value: '0b' for OTG controller core .
11 —	Reserved
10 ULPII	ULPI Interrupt This bit will be set '1b' by hardware when there is an event completion in ULPI viewport. This bit is usable only if the controller support UPLI interface mode.
9 —	Reserved
8 SLI	DCSuspend When a controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used in device operation mode.
7 SRI	SOF Received When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1 ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Because the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	In host mode, this bit will be set every 125 µs and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it.
6 URI	USB Reset Received When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device operation mode.
5 AAI	Interrupt on Async Advance System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the n_USBCMD register. This status bit indicates the assertion of that interrupt source. Only used in host operation mode.
4 SEI	System Error This bit is will be set to '1b' when an Error response is seen to a read on the system interface.
3 FRI	Frame List Rollover The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USB_n_USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used in host operation mode.
2 PCI	Port Change Detect The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively.
1 UEI	USB Error Interrupt (USBERRINT) When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT) This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

64.8.1.21 Interrupt Enable (USBINTR)

Offset

Register	Offset
USBINTR	148h

Function

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt source is active. The USB Status register (n_USBSTS) still shows interrupt sources even if they are disabled by the n_USBINTR register, allowing polling of interrupt events by the software.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved						TIE1	TIE0	Reserved				UPIE	UAIE	Reserved	NAKE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved					Reserved	Reserved	SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25 TIE1	General Purpose Timer #1 Interrupt Enable When this bit is one and the TI1 bit in n_USBSTS register is a one the controller will issue an interrupt.
24 TIE0	General Purpose Timer #0 Interrupt Enable When this bit is one and the TI0 bit in n_USBSTS register is a one the controller will issue an interrupt.
23-20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
19 UPIE	USB Host Periodic Interrupt Enable When this bit is one, and the UPI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
18 UAIE	USB Host Asynchronous Interrupt Enable When this bit is one, and the UAI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
17 —	Reserved
16 NAKE	NAK Interrupt Enable When this bit is one and the NAKI bit in n_USBSTS register is a one the controller will issue an interrupt.
15-11 —	These bits are reserved and should be set to zero.
10 —	Reserved
9 —	Reserved
8 SLE	Sleep Interrupt Enable When this bit is one and the SLI bit in n_n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
7 SRE	SOF Received Interrupt Enable When this bit is one and the SRI bit in n_USBSTS register is a one the controller will issue an interrupt.
6 URE	USB Reset Interrupt Enable When this bit is one and the URI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
5 AAE	Async Advance Interrupt Enable When this bit is one and the AAI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
4 SEE	System Error Interrupt Enable When this bit is one and the SEI bit in n_USBSTS register is a one the controller will issue an interrupt.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Only used in host operation mode.
3 FRE	Frame List Rollover Interrupt Enable When this bit is one and the FRI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
2 PCE	Port Change Detect Interrupt Enable When this bit is one and the PCI bit in n_USBSTS register is a one the controller will issue an interrupt.
1 UEE	USB Error Interrupt Enable When this bit is one and the UEI bit in n_USBSTS register is a one the controller will issue an interrupt.
0 UE	USB Interrupt Enable When this bit is one and the UI bit in n_USBSTS register is a one the controller will issue an interrupt.

64.8.1.22 USB Frame Index (FRINDEX)

Offset

Register	Offset
FRINDEX	14Ch

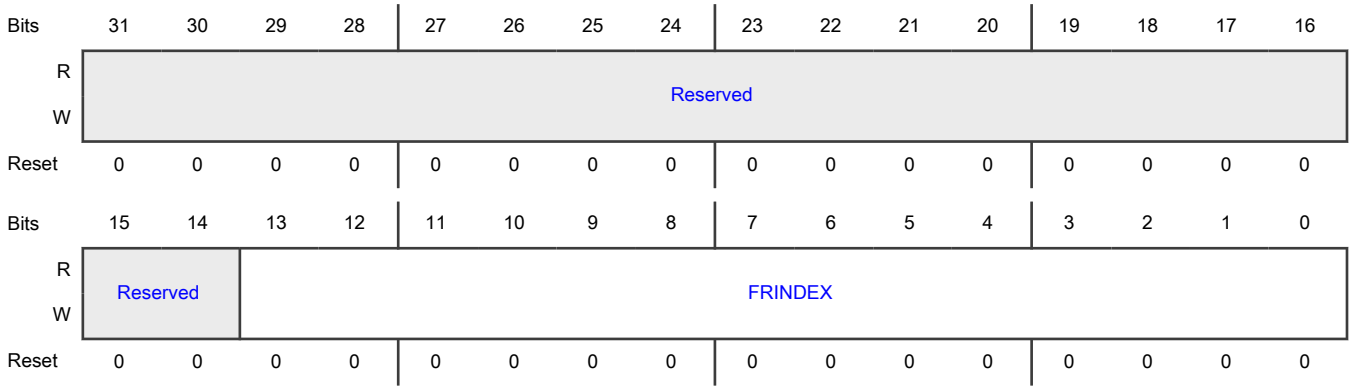
Function

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the n_USBCMD register.

This register must be written as a DWord. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be increment (that is, SOF for 125 us micro-frame.).

Diagram



Fields

Field	Function
31-14 —	Reserved
13-0 FRINDEX	Frame Index The value, in this register, increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe. The bit field values description below is represented as (Frame List Size) Number Elements N. 00_0000_0000_0000b - (1024) 12 00_0000_0000_0001b - (512) 11 00_0000_0000_0010b - (256) 10 00_0000_0000_0011b - (128) 9 00_0000_0000_0100b - (64) 8 00_0000_0000_0101b - (32) 7 00_0000_0000_0110b - (16) 6 00_0000_0000_0111b - (8) 5

64.8.1.23 Device Address (DEVICEADDR)

Offset

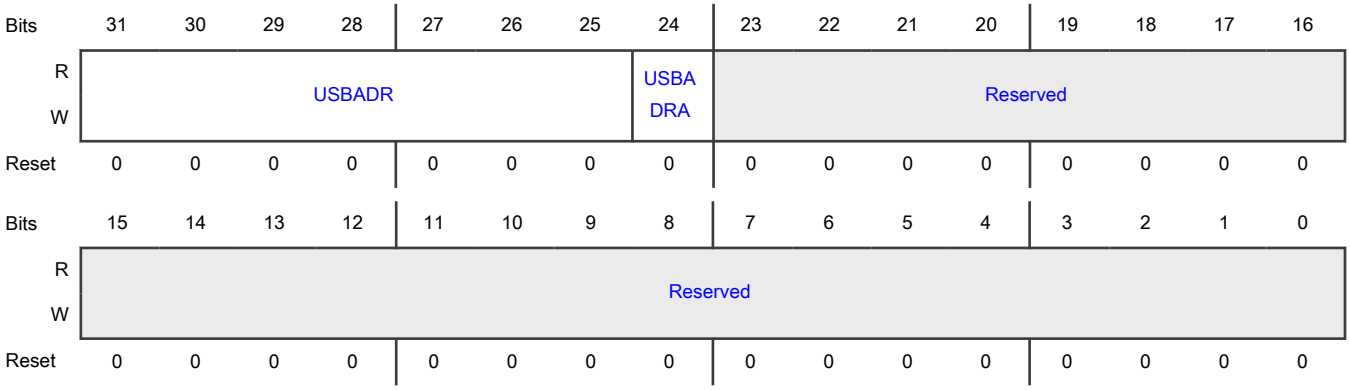
Register	Offset
DEVICEADDR	154h

Function

Device Controller only

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

Diagram



Fields

Field	Function
31-25 USBADR	Device Address These bits correspond to the USB device address
24 USBADRA	Device Address Advance When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a '1' at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: • IN is ACKed to endpoint 0. (USBADR is updated from staging register). • OUT/SETUP occur to endpoint 0. (USBADR is not updated). • Device Reset occurs (USBADR is reset to 0).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
23-0 —	- Reserved

64.8.1.24 Frame List Base Address (PERIODICLISTBASE)

Offset

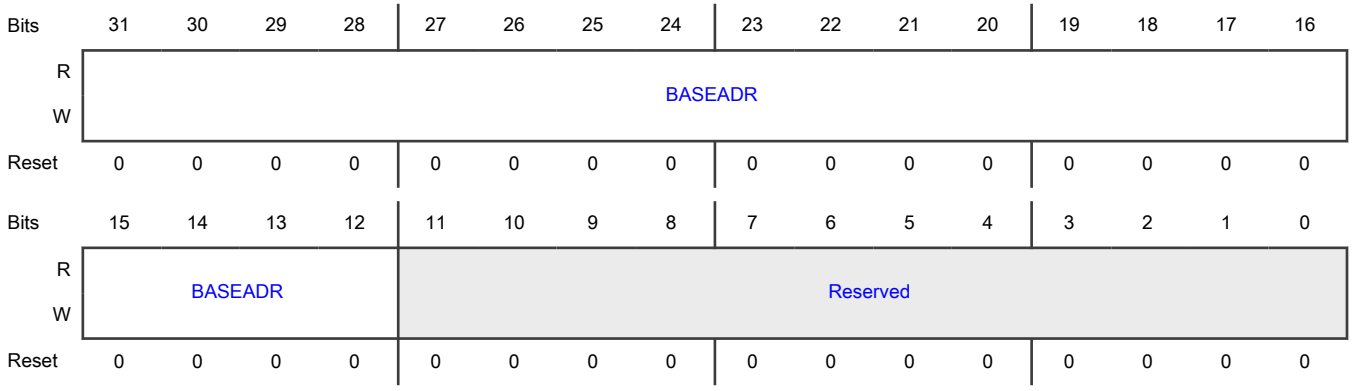
Register	Offset
PERIODICLISTBASE	154h

Function

Host Controller only

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (USB_n_FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Diagram



Fields

Field	Function
31-12 BASEADR	Base Address (Low) These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
11-0 —	- Reserved

64.8.1.25 Next Asynch. Address (ASYNCLISTADDR)

Offset

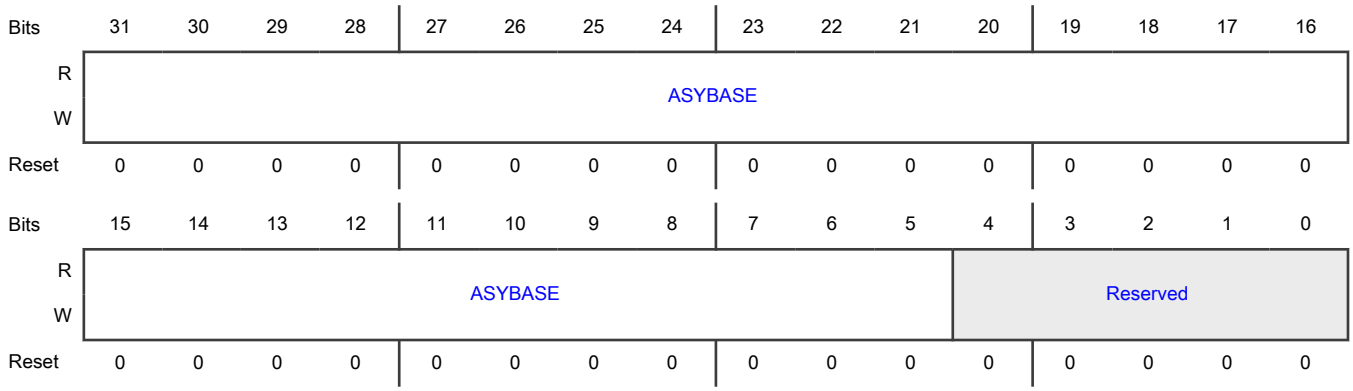
Register	Offset
ASYNCLISTADDR	158h

Function

Host Controller only

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Diagram



Fields

Field	Function
31-5 ASYBASE	Link Pointer Low (LPL) These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). Only used by the host controller.

Table continues on the next page...

Table continued from the previous page...

Field	Function
4-0 —	- Reserved

64.8.1.26 Endpoint List Address (ENDPTLISTADDR)

Offset

Register	Offset
ENDPTLISTADDR	158h

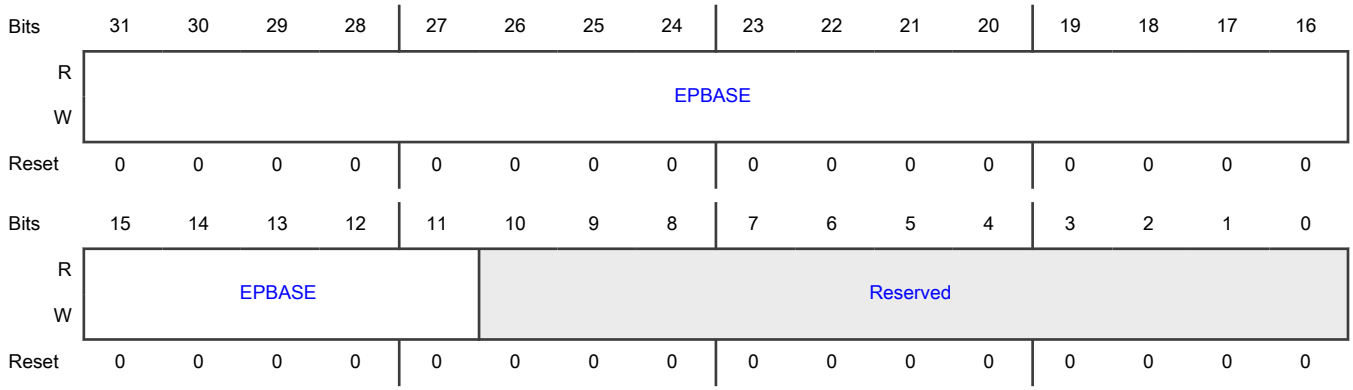
Function

Device Controller only

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

Diagram



Fields

Field	Function
31-11 EPBASE	Endpoint List Pointer (Low) These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Head (QH) (that is, one queue head per endpoint & direction).
10-0 —	- Reserved

64.8.1.27 Programmable Burst Size (BURSTSIZE)

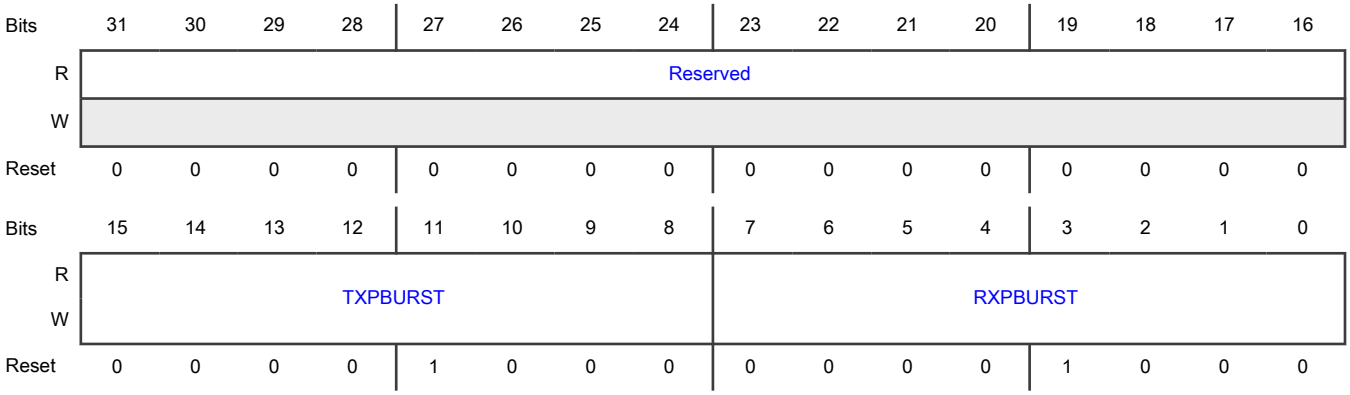
Offset

Register	Offset
BURSTSIZE	160h

Function

This register is used to control the burst size used during data movement on the AHB master interface. This register is ignored if AHBBRST bits in SBUSCFG register is non-zero value.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 TXPBURST	Programmable TX Burst Size Default value is determined by TXBURST bits in n_HWTXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
7-0 RXPBURST	Programmable RX Burst Size Default value is determined by TXBURST bits in n_HWRXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

64.8.1.28 TX FIFO Fill Tuning (TXFILLTUNING)

Offset

Register	Offset
TXFILLTUNING	164h

Function

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

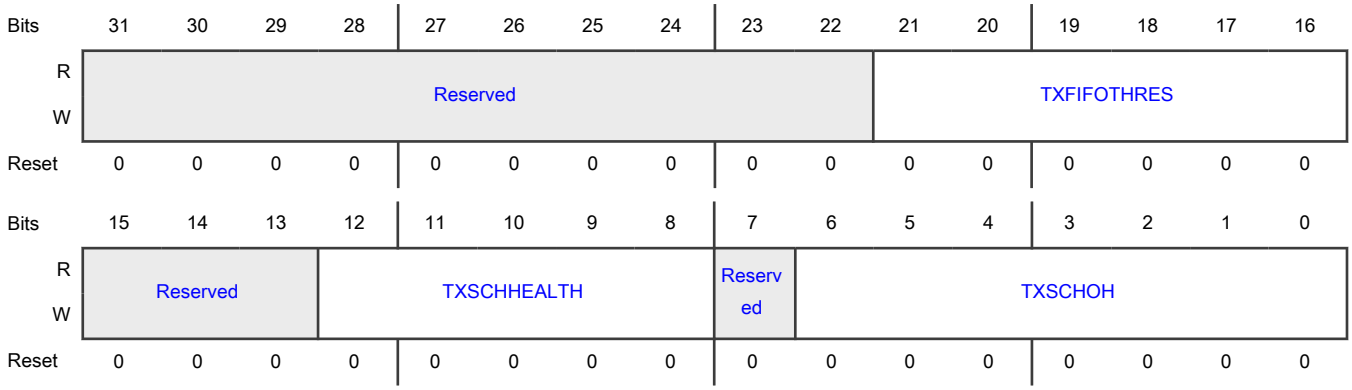
- T_0 = Standard packet overhead
- T_1 = Time to send data payload
- T_{ff} = Time to fetch packet into TX FIFO up to specified level.
- T_s = Total Packet Flight Time (send-only) packet
- $T_s = T_0 + T_1$
- T_p = Total Packet Time (fetch and send) packet
- $T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the n_TSCHHEALTH (T_{ff}) described below.

NOTE

The reset value could vary from instance to instance. See the detail in bit field description and ignore reset value in summary table in this case!

Diagram



Fields

Field	Function
31-22 —	Reserved
21-16 TXFIFOTHRES	FIFO Burst Threshold This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USB_n_USBMODE register is set.
15-13 —	Reserved
12-8 TXSCHHEALTH	Scheduler Health Counter This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7 —	Reserved
6-0 TXSCHOH	Scheduler Overhead This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode. Default value is '08h' for OTG controller core .

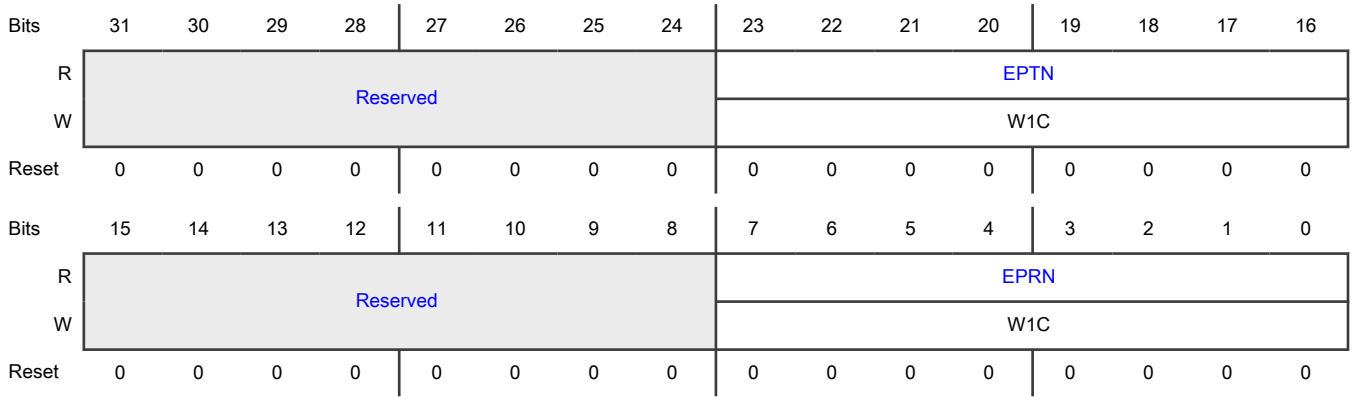
64.8.1.29 Endpoint NAK (ENDPTNAK)

Offset

Register	Offset
ENDPTNAK	178h

Function
Configures RX and TX endpoint NAKs.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 EPTN	TX Endpoint NAK Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7
15-8 —	Reserved
7-0 EPRN	RX Endpoint NAK Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7

64.8.1.30 Endpoint NAK Enable (ENDPTNAKEN)

Offset

Register	Offset
ENDPTNAKEN	17Ch

Function

Enables RX and TX endpoint NAKs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								EPTNE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								EPRNE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-16 EPTNE	TX Endpoint NAK Enable Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7
15-8 —	Reserved
7-0 EPRNE	RX Endpoint NAK Enable Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7

64.8.1.31 Configure Flag (CONFIGFLAG)

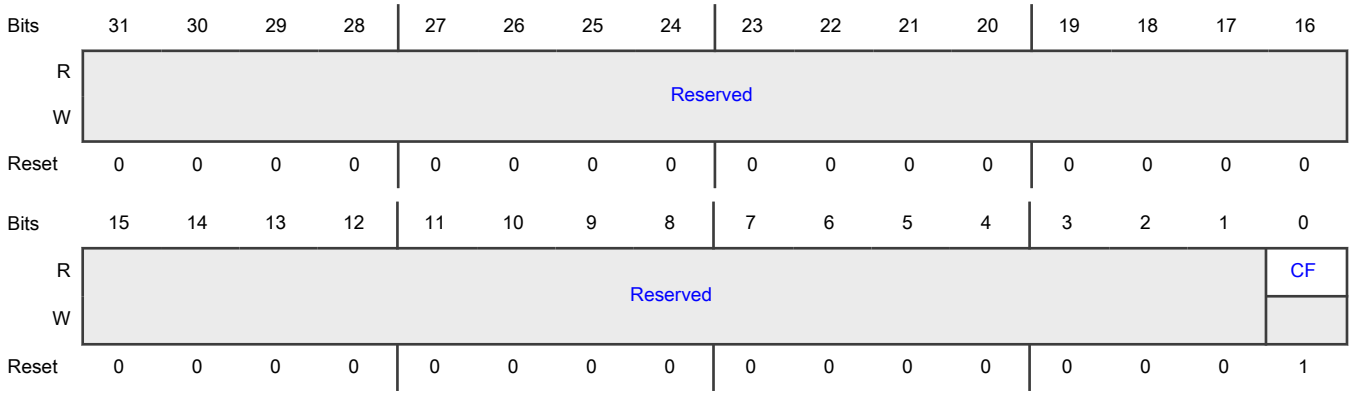
Offset

Register	Offset
CONFIGFLAG	180h

Function

Controls port-routing control logic.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 CF	Configure Flag Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. 0b - Port routing control logic default-routes each port to an implementation dependent classic host controller 1b - Port routing control logic default-routes all ports to this host controller

64.8.1.32 Port Status & Control (PORTSC1)

Offset

Register	Offset
PORTSC1	184h

Function

Host Controller

A host controller could implement one to eight port status and control registers. The number is determined by N_PORTs bits in HWSPARAMs register (see HCSPARAMS). Software could read this parameter register to determine how many ports need service.

All controller cores on this product are Single-Port, so there is only one port status and control register for each controller core.

This register is only reset by power on reset or controller core reset. The initial conditions of a port are:

- No device connected
- Port disabled

If the port supports power control, this state remains until port power is supplied (by software).

Device Controller

A controller operating in device mode has only port register one (PORTSC1) and it does not support power control in that mode. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTS_1		STS	PTW	PSPD		PTS_2	PFSC	PHCD	WKOC	WKDC	WKCN	PTC			
W																
Reset	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
W											W1C		W1C		W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31-30 PTS_1	Parallel Transceiver Select Parallel Transceiver Select (bit25, bit31, bit30). For OTG core, it's Read-Only. For Host1/Host2/Host3 core, it's Read/Write. **NOTE** All USB port interface modes are listed in this field description, but not all are supported. For detail feature of each controller core, see Features . The behaviour is unknown when unsupported interface mode is selected. Bit field {bit25, bit31, bit30}: This field defines bits 1:0 of the following values. PORTSC1[PTS_2] defines bit 2. 000b - UTMI/UTMI+ 001b - Reserved 010b - ULPI 011b - Parallel Transceiver Select 100b - HSIC
29 STS	Serial Transceiver Select Serial Interface Engine can be used in combination with UTMI+/ULPI physical interface to provide FS/LS signaling instead of the parallel interface signals. When this bit is set '1b', serial interface engine will be used instead of parallel interface signals. This bit has no effect unless PTS bits is set to select UTMI+/ULPI interface.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The Serial/USB1.1 PHY/IC-USB will use the serial interface engine for FS/LS signaling regardless of this bit value. 0b - Parallel Interface signals is selected 1b - Serial Interface Engine is selected
28 PTW	Parallel Transceiver Width - Read/Write Parallel Transceiver Width This bit has no effect if serial interface engine is used. 0b - Select the 8-bit UTMI interface [60 MHz] 1b - Select the 16-bit UTMI interface [30 MHz]
27-26 PSPD	Port Speed This register field indicates the speed at which the port is operating. 00b - Full Speed 01b - Low Speed 10b - High Speed 11b - Undefined
25 PTS_2	Parallel Transceiver Select See description at bits 31-30 Bit field {bit25, bit31, bit30}: This field defines bit 2 of the following values. PORTSC1[PTS_1] defines bits 1:0. 000b - UTMI/UTMI+ 001b - Reserved 010b - ULPI 011b - Parallel Transceiver Select 100b - HSIC
24 PFSC	Port Force Full Speed Connect When this bit is set to '1b', the port will be forced to only connect at Full Speed, It disables the chirp sequence that allows the port to identify itself as High Speed. 0b - Normal operation 1b - Forced to full speed
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD) When this bit is set to '1b', the PHY clock is disabled. Reading this bit will indicate the status of the PHY clock.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend when the device is not running (USBCMD Run/Stop=0b) or the host has signalled suspend (PORTSC1 SUSPEND=1b). PHY Low power suspend will be cleared automatically when the host initials resume. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. 0b - Enable PHY clock 1b - Disable PHY clock
22 WKOC	Wake on Over-current Enable (WKOC_E) Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PORTSC1) is zero.
21 WKDC	Wake on Disconnect Enable (WKDSCNNT_E) Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PORTSC1) is zero or in device mode.
20 WKCN	Wake on Connect Enable (WKCNNT_E) Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PORTSC1) is zero or in device mode.
19-16 PTC	Port Test Control Refer to Port test mode for the operational model for using these test modes and the USB Specification Revision 2.0, Chapter 7 for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **NOTE** Low speed operations are not supported as a peripheral device. Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000b - TEST_MODE_DISABLE 0001b - J_STATE 0010b - K_STATE 0011b - SE0 (host) / NAK (device)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0100b - Packet 0101b - FORCE_ENABLE_HS 0110b - FORCE_ENABLE_FS 0111b - FORCE_ENABLE_LS 1000b-1111b - Reserved
15-14 PIC	Port Indicator Control Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if Port Power (PORTSC1) is zero. 00b - Port indicators are off 01b - Amber 10b - Green 11b - Undefined
13 PO	Port Owner This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not supported in all controller cores, therefore this bit will always be 0.
12 PP	Port Power The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: • PPC = 0b, PP = 1b (Read Only) - Host controller does not have port power control switches. Each port is hard-wired to power. • PPC = 1b, PP = 1b/0b (Read/Write) - OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitional by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in all controller cores (PPC = 1).
11-10 LS	Line Status These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary. 00b - SE0 01b - K-state 10b - J-state 11b - Undefined
9 HSP	High-Speed Port When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. **NOTE** HSP is redundant with PSPD(bit 27, 26) but remained for compatibility.
8 PR	Port Reset In Host Mode: Read/Write. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. This field is zero if Port Power (PORTSC1) is zero. 0b - Port is not in reset 1b - Port is in reset
7 SUSP	Suspend In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State • 0x Disable • 10 Enable • 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, Port enabled bit is a zero) the results are undefined. This field is zero if Port Power (PORTSC1) is zero in host mode. In Device Mode: Read Only. In device mode this bit is a read only status bit. 0b - Port not in suspend state 1b - Port in suspend state
6 FPR	Force Port Resume In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation, clear the bit the port control state switches to HS or FS idle. This field is zero if Port Power (PORTSC1) is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit will be cleared because a K-to-J transition detected, the Port Change Detect bit in the USBSTS register is also set to one. 0b - No resume (K-state) detected/driven on port 1b - Resume detected/driven on port
5 OCC	Over-current Change This bit is set '1b' by hardware when there is a change to Over-current Active. Software can clear this bit by writing a one to this bit position.

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 OCA	Over-Current Active This bit will automatically transition from one to zero when the over current condition is removed. 0b - This port does not have an over-current condition 1b - This port currently has an over-current condition
3 PEC	Port Enable/Disable Change In Host Mode: For the root hub, this bit is set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PORTSC1) is zero. In Device mode: The device port is always enabled, so this bit is always '0b'. 0b - No change 1b - Port enabled/disabled status has changed
2 PE	Port Enabled/Disabled In Host Mode: This bit is reset to '0b'. Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. **NOTE** The bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power (PORTSC1) is zero in host mode. In Device Mode: The device port is always enabled, so this bit is always '1b'. 0b - Disable 1b - Enable
1 CSC	Connect Status Change In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the

Table continues on the next page...

Table continued from the previous page...

Field	Function
	changed condition, hub hardware will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PORTSC1) is zero in host mode. In Device Mode: This bit is undefined in device controller mode. 0b - No change 1b - Change in current connect status
0 CCS	Current Connect Status In Host Mode: This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PORTSC1) is zero in host mode. In Device Mode: A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0b - In Host mode: No device is present. In Device mode: Not attached 1b - In Host mode: Device is present on port. In Device mode: Attached

64.8.1.33 On-The-Go Status & Control (OTGSC)

Offset

Register	Offset
OTGSC	1A4h

Function

This register is available only in OTG controller core. It has four sections:

- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

The status inputs are debounced using a 1 ms time constant. Values on the status inputs that do not persist for more than 1 ms does not cause an update of the status input register, or cause an OTG interrupt.

See also USBMODE register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	DPIE	EN_1MS	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	Reserved	DPIS	STATUS_...	BSEIS	BSVIS	ASVIS	AVVIS	IDIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	u	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	DPS	TOG_1MS	BSE	BSV	ASV	AVV	ID	Reserved		IDPU	DP	OT	Reserved	VC	VD
W																
Reset	0	0	u	1	0	0	u	0	0	0	1	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 DPIE	Data Pulse Interrupt Enable
29 EN_1MS	1 Millisecond Timer Interrupt Enable
28 BSEIE	B Session End Interrupt Enable Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable Setting this bit enables the USB ID interrupt.
23 —	Reserved
22 DPIS	Data Pulse Interrupt Status

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC1(0)[PP] = 0. Software must write a one to clear this bit.
21 STATUS_1MS	1 Millisecond Timer Interrupt Status This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit.
19 BSVIS	B Session Valid Interrupt Status This bit is set when VBus has either risen above or fallen below the B session valid threshold. Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status This bit is set when VBus has either risen above or fallen below the A session valid threshold. Software must write a one to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status This bit is set when VBus has either risen above or fallen below the VBus valid threshold on an A device. Software must write a one to clear this bit.
16 IDIS	USB ID Interrupt Status This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.
15 —	Reserved
14 DPS	Data Bus Pulsing Status A '1' indicates data bus pulsing is being detected on the port.
13 TOG_1MS	1 Millisecond Timer Toggle This bit toggles once per millisecond.
12 BSE	B Session End Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid Indicates VBus is above the B session valid threshold.

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 ASV	A Session Valid Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID 0b - A device 1b - B device
7-6 —	Reserved
5 IDPU	ID Pullup This bit provide control over the ID pull-up resistor; When this bit is 0, the ID input will not be sampled. 0b - Off 1b - On
4 DP	Data Pulsing Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 —	Reserved
1 VC	VBUS Charge Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS Discharge Setting this bit causes VBus to discharge through a resistor.

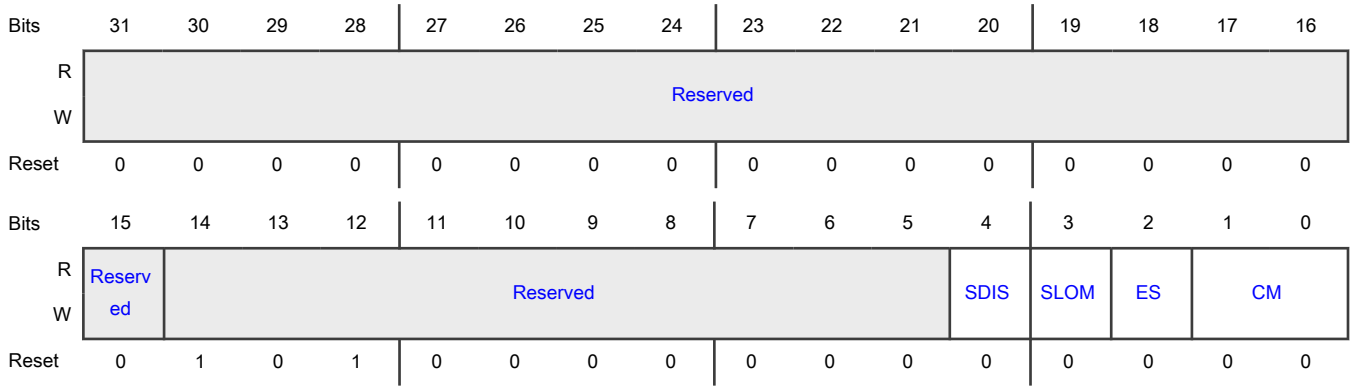
64.8.1.34 USB Device Mode (USBMODE)

Offset

Register	Offset
USBMODE	1A8h

Function
Configures USB device operation mode.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 —	Reserved
14-5 —	Reserved
4 SDIS	Stream Disable Mode Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. **NOTE** In High Speed Mode, all packets received are responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. **NOTE** Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TXFILLTUNING and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature. **NOTE** The use of this feature substantially limits of the overall USB performance that can be achieved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Inactive 1b - Active
3 SLOM	Setup Lockout Mode In device mode, this bit controls behavior of the setup lock mechanism. See Control endpoint operation model. 0b - Setup Lockouts On (default); 1b - Setup Lockouts Off. DCD requires use of Setup Data Buffer Tripwire in USBCMD.
2 ES	Endian Select This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. 0b - Little Endian 1b - Big Endian
1-0 CM	Controller Mode Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. For OTG controller core, reset value is '00b'. 00b - Idle [Default for combination host/device] 01b - Reserved 10b - Device Controller [Default for device only controller] 11b - Host Controller [Default for host only controller]

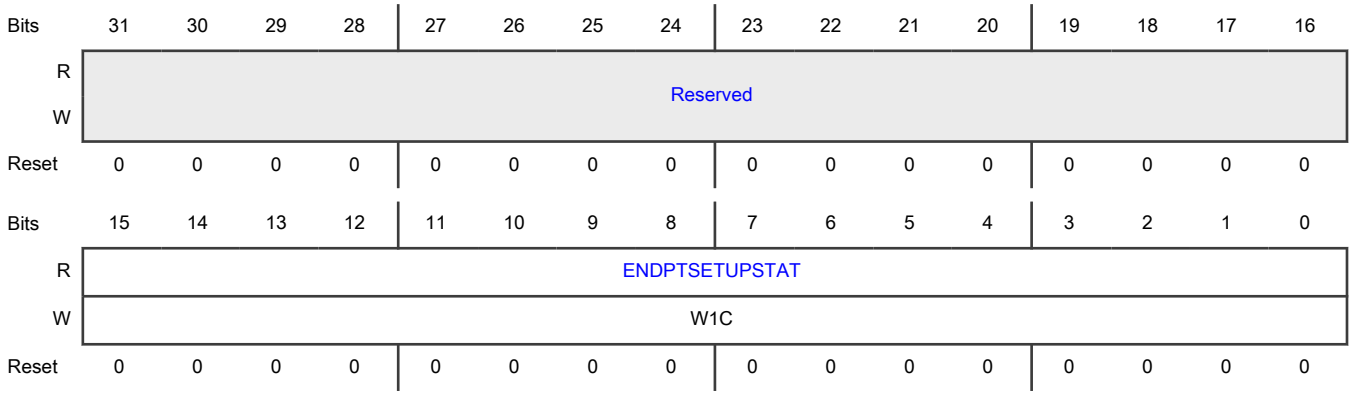
64.8.1.35 Endpoint Setup Status (ENDPTSETUPSTAT)

Offset

Register	Offset
ENDPTSETUPSTAT	1ACh

Function
Indicates endpoint setup status.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 ENDPTSETUPSTAT	Setup Endpoint Status For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See Managing endpoints in the Device Operational Model. This register is only used in device mode.

64.8.1.36 Endpoint Prime (ENDPTPRIME)

Offset

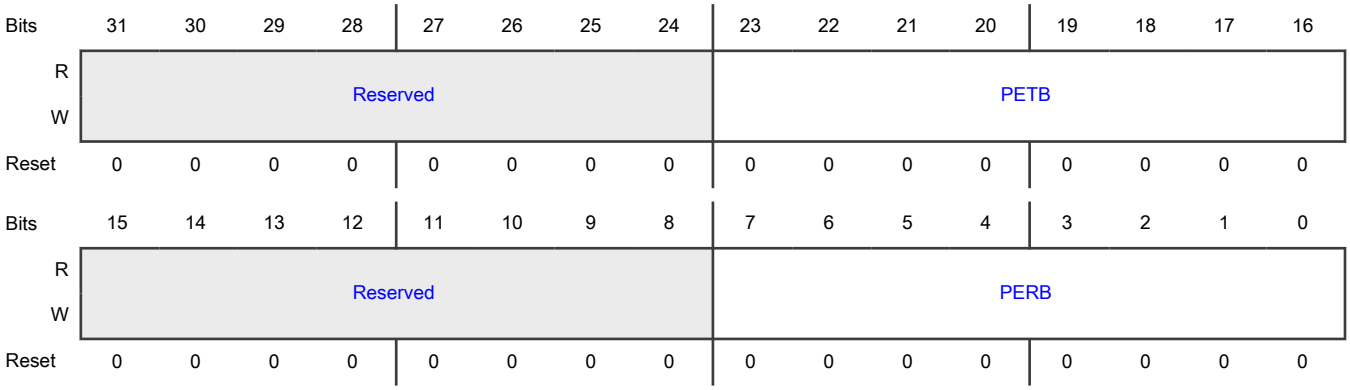
Register	Offset
ENDPTPRIME	1B0h

Function

This register is only used in device mode.

When software sets the prime bit for a given endpoint, the device controller loads the transfer descriptor, pointed to by the queue head, such that the endpoint is ready to transmit or receive when the host sends a request (IN/OUT token). The endpoint will NAK all requests from the host until the endpoint is primed. The controller will automatically re-prime the endpoint with a new transfer descriptor when one is found via the next_dtd pointer of the current transfer descriptor. Hence, the prime bit must only be set by software when a descriptor is added to the queue head.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 PETB	Prime Endpoint Transmit Buffer For each endpoint a corresponding bit is used to request that a buffer is prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[N] - Endpoint #N, N is in 0..7
15-8 —	Reserved
7-0 PERB	Prime Endpoint Receive Buffer For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PERB[N] - Endpoint #N, N is in 0..7

64.8.1.37 Endpoint Flush (ENDPTFLUSH)

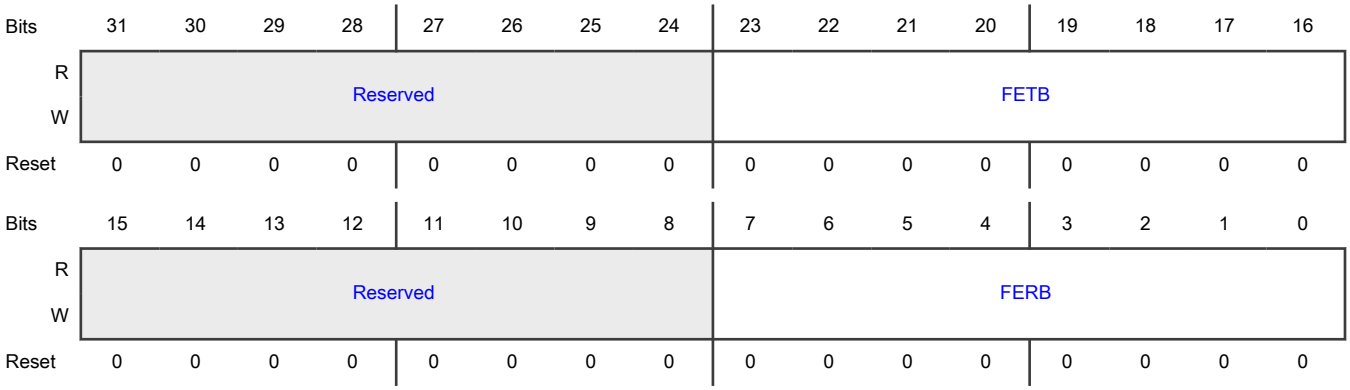
Offset

Register	Offset
ENDPTFLUSH	1B4h

Function

This register is only used in device mode.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 FETB	Flush Endpoint Transmit Buffer Writing one to a bit(s) in this register causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FETB[N] - Endpoint #N, N is in 0..7
15-8 —	Reserved
7-0 FERB	Flush Endpoint Receive Buffer Writing one to a bit(s) causes the assocUOGiated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FERB[N] - Endpoint #N, N is in 0..7\

64.8.1.38 Endpoint Status (ENDPTSTAT)

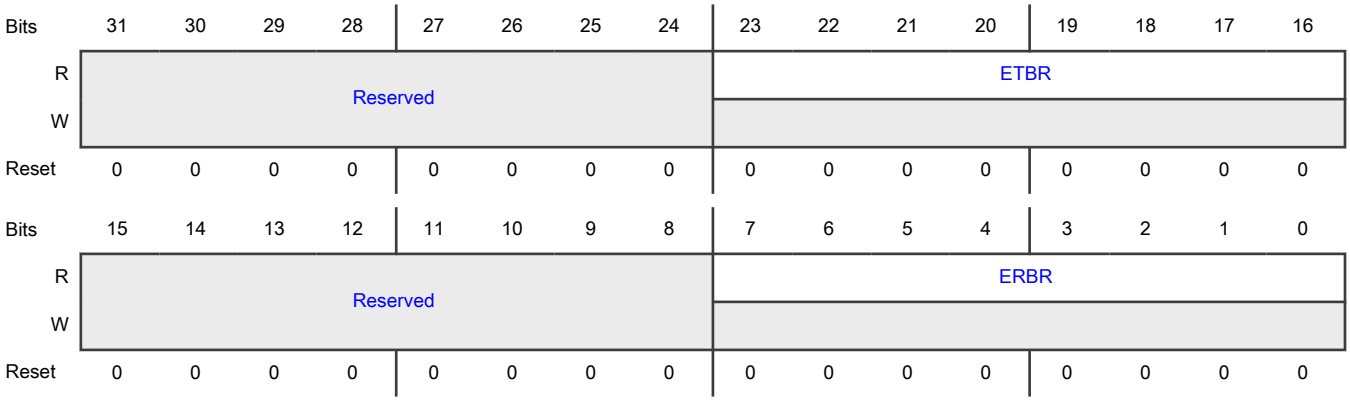
Offset

Register	Offset
ENDPTSTAT	1B8h

Function

This register is only used in device mode.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 ETBR	Endpoint Transmit Buffer Ready One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There is always a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[N] - Endpoint #N, N is in 0..7
15-8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 ERBR	Endpoint Receive Buffer Ready One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPRIME register. There is always a delay between setting a bit in the ENDPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[N] - Endpoint #N, N is in 0..7

64.8.1.39 Endpoint Complete (ENDPTCOMPLETE)

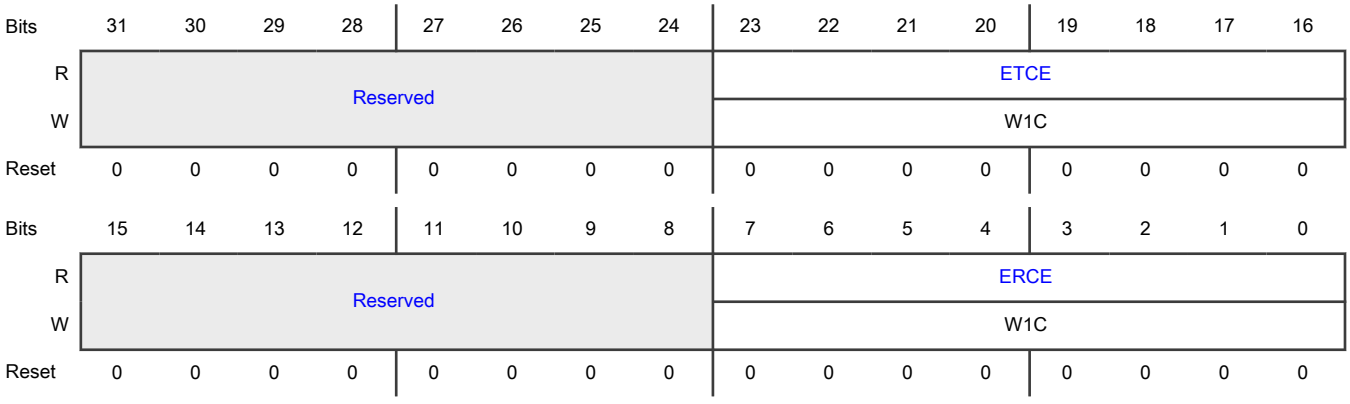
Offset

Register	Offset
ENDPTCOMPLETE	1BCh

Function

This register is only used in device mode.

Diagram



Fields

Field	Function
31-24	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
23-16 ETCE	Endpoint Transmit Complete Event Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the USBINT. Writing one clears the corresponding bit in this register. ETCE[N] - Endpoint #N, N is in 0..7
15-8 —	Reserved
7-0 ERCE	Endpoint Receive Complete Event Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the USBINT. Writing one clears the corresponding bit in this register. ERCE[N] - Endpoint #N, N is in 0..7

64.8.1.40 Endpoint Control 0 (ENDPTCTRL0)

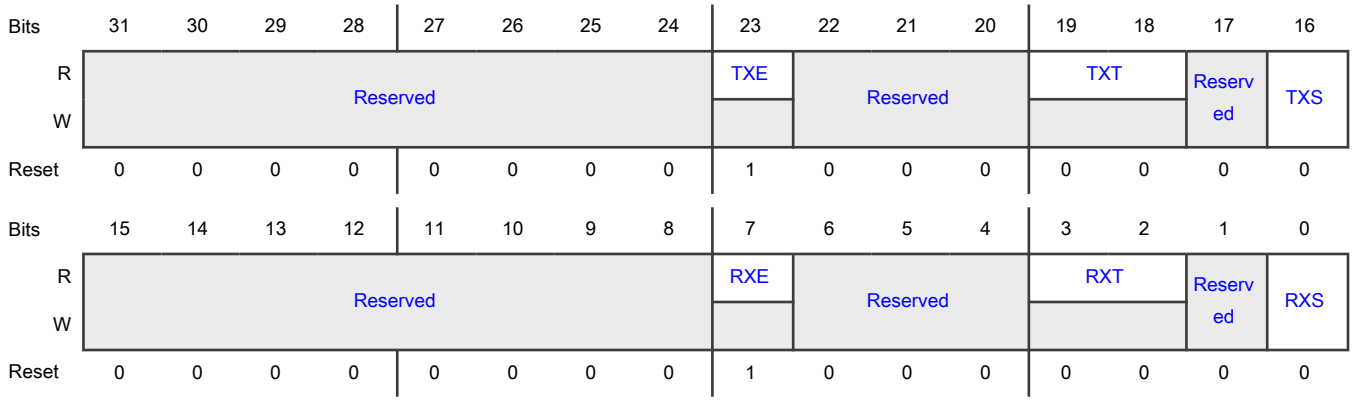
Offset

Register	Offset
ENDPTCTRL0	1C0h

Function

Every Device implements Endpoint 0 as a control endpoint.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable Endpoint0 is always enabled. 0b - Disabled 1b - Enabled
22-20 —	Reserved
19-18 TXT	TX Endpoint Type Endpoint 0 is fixed as a Control End Point. 00b - Control
17 —	Reserved
16 TXS	TX Endpoint Stall Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. **NOTE** There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable Endpoint0 is always enabled. 0b - Disabled 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
6-4 —	Reserved
3-2 RXT	RX Endpoint Type Endpoint 0 is fixed as a Control End Point. 00b - Control
1 —	Reserved
0 RXS	RX Endpoint Stall Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. **NOTE** There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the dcd software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled

64.8.1.41 Endpoint Control 1 (ENDPTCTRL1)

Offset

Register	Offset
ENDPTCTRL1	1C4h

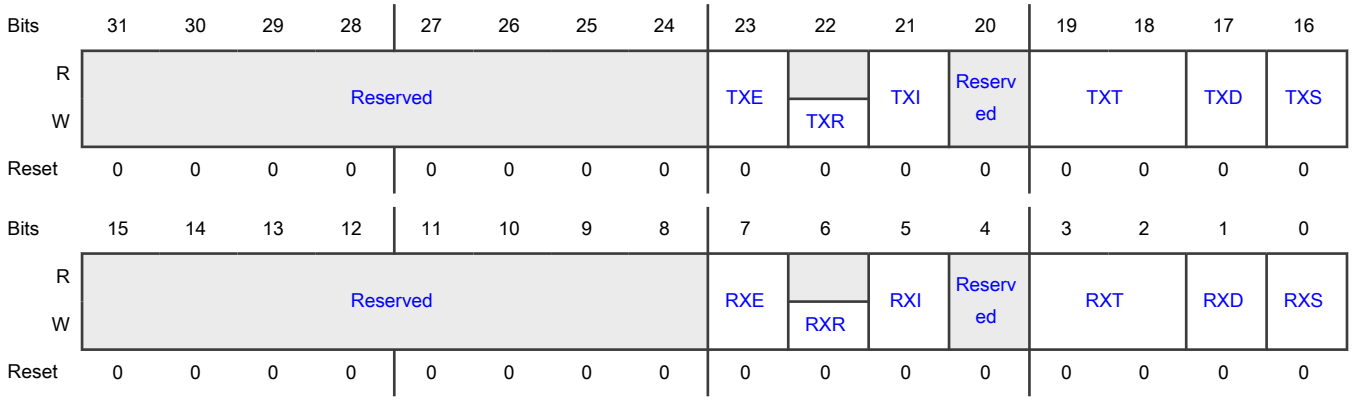
Function

This is endpoint control register for endpoint 1 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
22 TXR	TX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. 1b - Reset PID sequence
21 TXI	TX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
20 —	Reserved
19-18 TXT	TX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 TXD	TX Endpoint Data Source Should always be written as 0. 0b - Dual Port Memory Buffer/DMA Engine
16 TXS	TX Endpoint Stall This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
6 RXR	RX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device. 1b - Reset PID sequence
5 RXI	RX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0b - Disabled 1b - Enabled
4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3-2 RXT	RX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
1 RXD	RX Endpoint Data Sink Should always be written as zero. 0b - Dual Port Memory Buffer/DMA Engine
0 RXS	RX Endpoint Stall This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled

64.8.1.42 Endpoint Control 2 (ENDPTCTRL2)

Offset

Register	Offset
ENDPTCTRL2	1C8h

Function

This is endpoint control register for endpoint 2 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
22 TXR	TX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. 1b - Reset PID sequence
21 TXI	TX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
20 —	Reserved
19-18	TX Endpoint Type

Table continues on the next page...

Table continued from the previous page...

Field	Function
TXT	00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
17 TXD	TX Endpoint Data Source Should always be written as 0. 0b - Dual Port Memory Buffer/DMA Engine
16 TXS	TX Endpoint Stall This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
6 RXR	RX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device. 1b - Reset PID sequence
5	RX Data Toggle Inhibit

Table continues on the next page...

Table continued from the previous page...

Field	Function
RXI	This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0b - Disabled 1b - Enabled
4 —	Reserved
3-2 RXT	RX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
1 RXD	RX Endpoint Data Sink Should always be written as zero. 0b - Dual Port Memory Buffer/DMA Engine
0 RXS	RX Endpoint Stall This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled

64.8.1.43 Endpoint Control 3 (ENDPTCTRL3)

Offset

Register	Offset
ENDPTCTRL3	1CCh

Function

This is endpoint control register for endpoint 3 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
22 TXR	TX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. 1b - Reset PID sequence
21 TXI	TX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
19-18 TXT	TX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
17 TXD	TX Endpoint Data Source Should always be written as 0. 0b - Dual Port Memory Buffer/DMA Engine
16 TXS	TX Endpoint Stall This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
6 RXR	RX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device. 1b - Reset PID sequence

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 RXI	RX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0b - Disabled 1b - Enabled
4 —	Reserved
3-2 RXT	RX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
1 RXD	RX Endpoint Data Sink Should always be written as zero. 0b - Dual Port Memory Buffer/DMA Engine
0 RXS	RX Endpoint Stall This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled

64.8.1.44 Endpoint Control 4 (ENDPTCTRL4)

Offset

Register	Offset
ENDPTCTRL4	1D0h

Function

This is endpoint control register for endpoint 4 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE		TXI	Reserved	TXT		TXD	TXS
W										TXR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RXE		RXI	Reserved	RXT		RXD	RXS
W										RXR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
22 TXR	TX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. 1b - Reset PID sequence

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 TXI	TX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
20 —	Reserved
19-18 TXT	TX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
17 TXD	TX Endpoint Data Source Should always be written as 0. 0b - Dual Port Memory Buffer/DMA Engine
16 TXS	TX Endpoint Stall This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable An endpoint should be enabled only after it has been configured.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled 1b - Enabled
6 RXR	RX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device. 1b - Reset PID sequence
5 RXI	RX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0b - Disabled 1b - Enabled
4 —	Reserved
3-2 RXT	RX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
1 RXD	RX Endpoint Data Sink Should always be written as zero. 0b - Dual Port Memory Buffer/DMA Engine
0 RXS	RX Endpoint Stall This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Endpoint OK 1b - Endpoint stalled

64.8.1.45 Endpoint Control 5 (ENDPTCTRL5)

Offset

Register	Offset
ENDPTCTRL5	1D4h

Function

This is endpoint control register for endpoint 5 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable An endpoint should be enabled only after it has been configured.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled 1b - Enabled
22 TXR	TX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. 1b - Reset PID sequence
21 TXI	TX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
20 —	Reserved
19-18 TXT	TX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
17 TXD	TX Endpoint Data Source Should always be written as 0. 0b - Dual Port Memory Buffer/DMA Engine
16 TXS	TX Endpoint Stall This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
6 RXR	RX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device. 1b - Reset PID sequence
5 RXI	RX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0b - Disabled 1b - Enabled
4 —	Reserved
3-2 RXT	RX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
1 RXD	RX Endpoint Data Sink Should always be written as zero. 0b - Dual Port Memory Buffer/DMA Engine
0 RXS	RX Endpoint Stall This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled

64.8.1.46 Endpoint Control 6 (ENDPTCTRL6)

Offset

Register	Offset
ENDPTCTRL6	1D8h

Function

This is endpoint control register for endpoint 6 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE		TXI	Reserved	TXT		TXD	TXS
W										TXR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RXE		RXI	Reserved	RXT		RXD	RXS
W										RXR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
22 TXR	TX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. 1b - Reset PID sequence
21 TXI	TX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
20 —	Reserved
19-18 TXT	TX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
17 TXD	TX Endpoint Data Source Should always be written as 0. 0b - Dual Port Memory Buffer/DMA Engine
16 TXS	TX Endpoint Stall This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
6 RXR	RX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device. 1b - Reset PID sequence
5 RXI	RX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0b - Disabled 1b - Enabled
4 —	Reserved
3-2 RXT	RX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
1 RXD	RX Endpoint Data Sink Should always be written as zero.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Dual Port Memory Buffer/DMA Engine
0 RXS	RX Endpoint Stall This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled

64.8.1.47 Endpoint Control 7 (ENDPTCTRL7)

Offset

Register	Offset
ENDPTCTRL7	1DCh

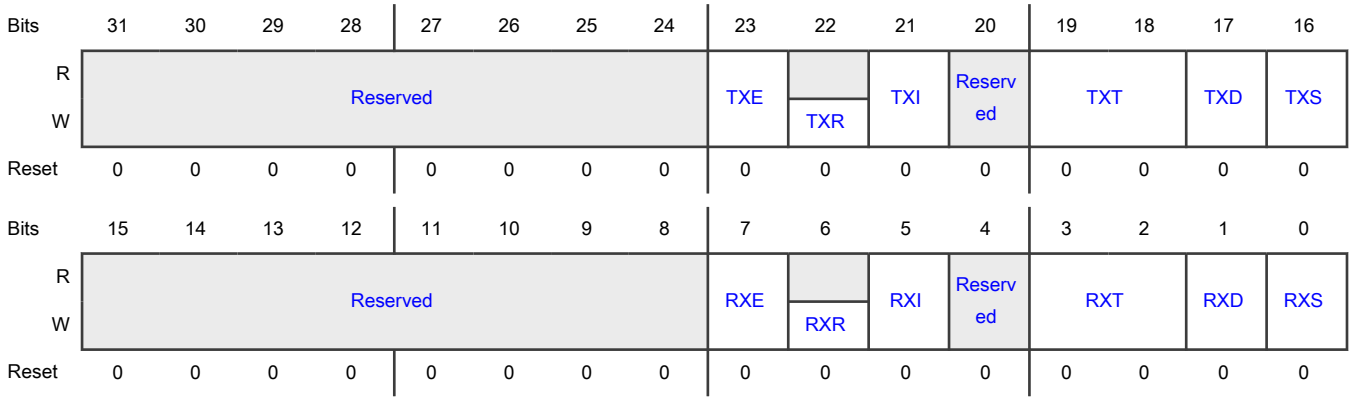
Function

This is endpoint control register for endpoint 7 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 TXE	TX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
22 TXR	TX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. 1b - Reset PID sequence
21 TXI	TX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
20 —	Reserved
19-18 TXT	TX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 TXD	TX Endpoint Data Source Should always be written as 0. 0b - Dual Port Memory Buffer/DMA Engine
16 TXS	TX Endpoint Stall This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled
15-8 —	Reserved
7 RXE	RX Endpoint Enable An endpoint should be enabled only after it has been configured. 0b - Disabled 1b - Enabled
6 RXR	RX Data Toggle Reset (WS) Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device. 1b - Reset PID sequence
5 RXI	RX Data Toggle Inhibit This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0b - Disabled 1b - Enabled
4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3-2 RXT	RX Endpoint Type 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
1 RXD	RX Endpoint Data Sink Should always be written as zero. 0b - Dual Port Memory Buffer/DMA Engine
0 RXS	RX Endpoint Stall This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit. 0b - Endpoint OK 1b - Endpoint stalled

Chapter 65
Ultra Secured Digital Host Controller (uSDHC)

65.1 Chip-specific uSDHC information

Table 535. Reference links to related information

Topic	Related module	Reference
Full description	uSDHC	uSDHC
Clocking		Clock distribution [1]
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

1. lp_osc is connected as the low power clock, in this device.

NOTE

See AHB Clock Control 2 (AHBCLKCTRL2) for clock control. Also see Peripheral Reset Control 2 (PRESETCTRL2) for reset control.

65.1.1 Module instances

This device has one instance of the uSDHC module, uSDHC0.

65.2 Overview

uSDHC provides the interface between the host system and the eMMC, SD card, and SDIO as shown in Figure 355. The module acts as a bridge, passing host bus transactions to the eMMC, SD card, and SDIO by sending commands and performing data accesses to/from the cards. It handles the SD card/SDIO/eMMC protocols at the transmission level.

65.2.1 Block diagram

The following figure illustrates the block diagram for uSDHC. Dual port SRAM is FIFO for write/read. For more information, see Data buffer.

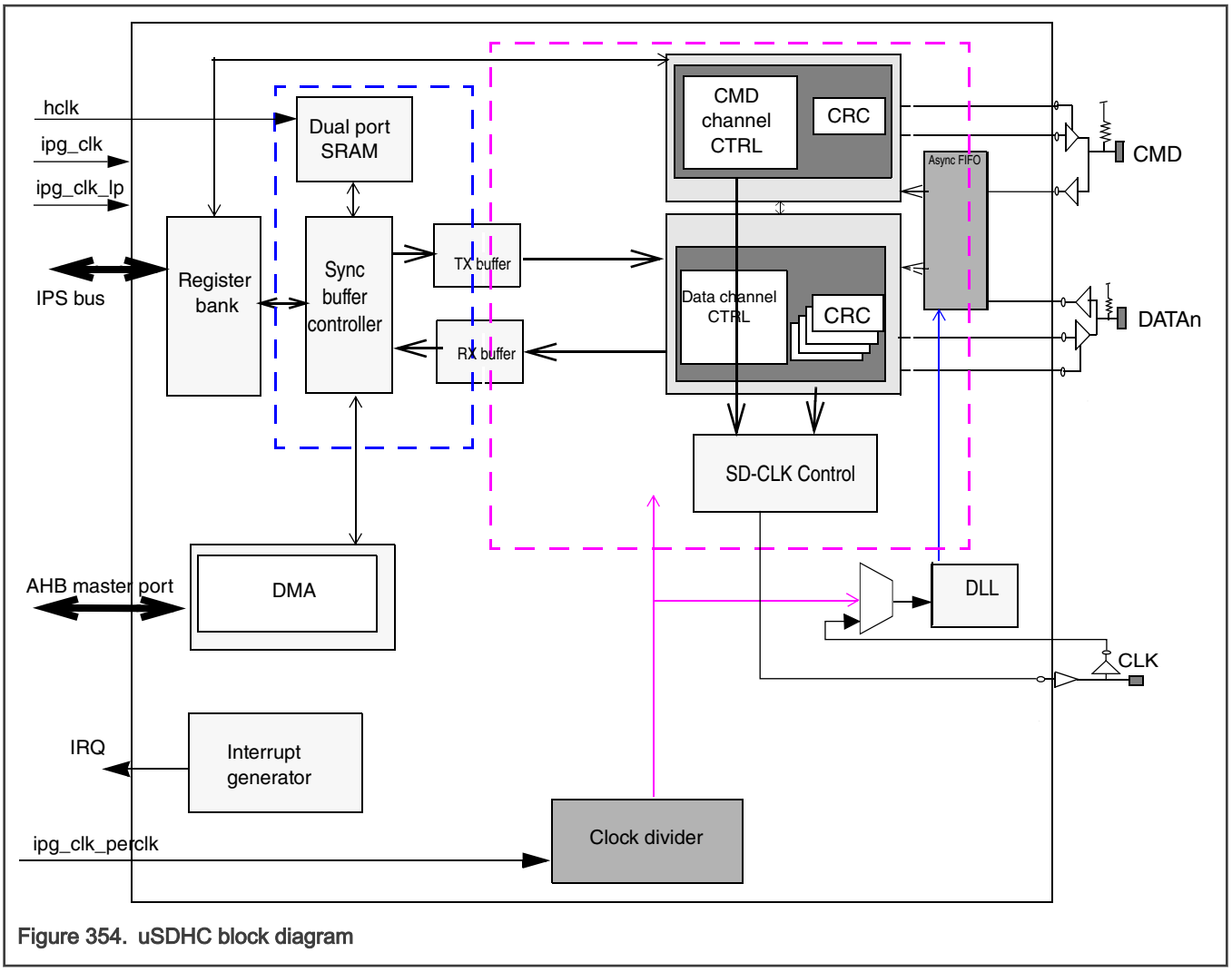


Figure 354. uSDHC block diagram

The figure below shows the System connection of uSDHC:

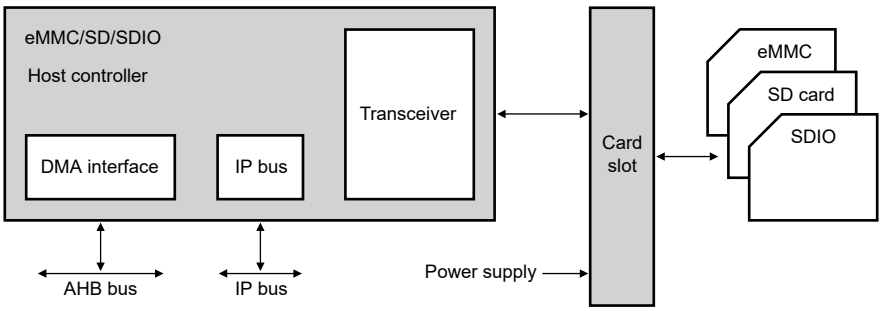


Figure 355. System connection of uSDHC

The following are brief descriptions of the cards supported by uSDHC:

- The Embedded MultiMediaCard (eMMC) is a universal low-cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous eMMC were based on a 7-pin serial bus with a single data pin, while the new high speed eMMC communication is based on an advanced 11-pin serial bus designed to operate in the low-voltage range.

- The Secure Digital Card (SD) is an evolution of the old eMMC technology. It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are backward compatible with the old eMMC (with some additions).
- Under the SD protocol, system connection can be categorized into memory card, I/O card, and combo card, which have both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI (Secure Digital Music Initiative) standard. The I/O card, which is also known as SDIO, provides high-speed data I/O with low-power consumption for mobile electronic devices.

65.2.2 Features

The list given below shows the features of the uSDHC module:

- Conforms to the SD Host Controller Standard Specification version 2.0/3.0
- Compatible with the eMMC System Specification version 4.2/4.3/4.4/4.41
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card
- Compatible with the SDIO Specification version 2.0/3.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, eMMC, MMC plus, and MMC RS cards
- For card bus clock frequency: See the product Data Sheet for the clock frequency of each supported mode.
- Supports 1-bit/4-bit SD and SDIO modes, and 1-bit/4-bit/8-bit eMMC modes
  - Up to 832 Mbps of data transfer for SDIO using four parallel data lines in the Single Data Rate (SDR) mode
  - Up to 400 Mbps of data transfer for SDIO using four parallel data lines in the Dual Data Rate (DDR) mode
  - Up to 832 Mbps of data transfer for SDXC cards using four parallel data lines in the Single Data Rate (SDR) mode
  - Up to 400 Mbps of data transfer for SDXC card using four parallel data lines in the Dual Data Rate (DDR) mode
  - Up to 416 Mbps of data transfer for eMMC using eight parallel data lines in the Single Data Rate (SDR) mode
  - Up to 832 Mbps of data transfer for eMMC using eight parallel data lines in the Dual Data Rate (DDR) mode
- Supports single block/multi-block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes; also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal DMA capabilities
- Supports voltage selection by configuring vendor-specific register bit
- Supports Advanced DMA to perform linked memory access

NOTE

This block can support all the above-listed speed mode and maximum data throughput. However, these may be specific to the device. See the corresponding chip-specific information or the device data sheet for accurate details.

65.3 Functional description

The following sections provide a brief functional description of the major system blocks, including the data buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock and reset manager, and clock generator.

65.3.1 Modes and operations

65.3.1.1 Data transfer modes

The uSDHC module can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- eMMC 1-bit
- eMMC 4-bit
- eMMC 8-bit
- Identification mode (up to 400 kHz)
- eMMC full-speed mode (up to 26 MHz)
- eMMC high-speed mode (up to 52 MHz)
- eMMC DDR mode (52 MHz both edges)
- SD card or SDIO full-speed mode (up to 25 MHz)
- SD card or SDIO high-speed mode (up to 50 MHz)
- SD card or SDIO UHS-I mode (up to 208 or 100 MHz in SDR mode, up to 50 MHz in the DDR mode)

NOTE

This block can support all the above listed speed mode and maximum clock frequency. However, these may be specific to the device. See the corresponding chip-specific information or the device data sheet for accurate details.

65.3.2 Data buffer

The uSDHC module uses one configurable data buffer to transfer data between the system bus (IP bus or advanced high-performance bus (AHB) bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock). The buffer is used as a temporary storage for transferring data between the host system and the card. The watermark levels for read and write are both configurable and can range between 1 to 128 words. The burst lengths for read and write are also configurable and can range between 1 to 31 words. The next figure provides the uSDHC buffer scheme as buffer control and buffer RAM wrapper.

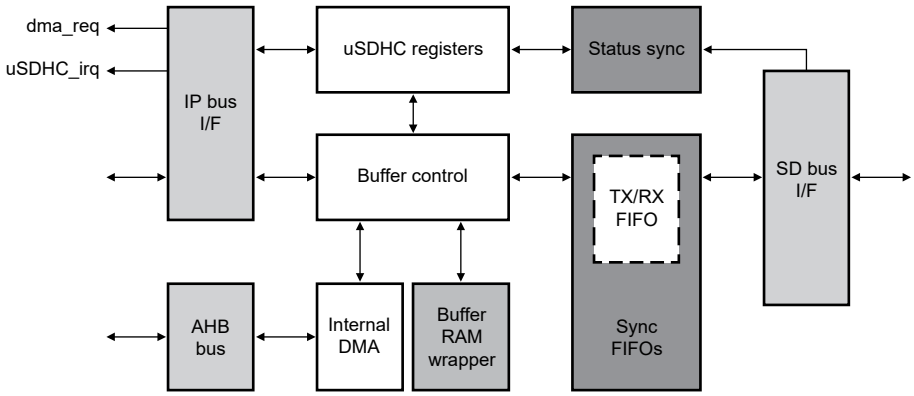


Figure 356. uSDHC buffer scheme

Here are 2 transfer modes to access the data buffer:

- CPU polling mode:
  - For a host-read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the host driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar. For more information on the process of writing operation, see Write operation sequence.
- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus.

For a read operation, when there are more words in the buffer than the amount set in WTMK_LVL[RD_WML], the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always the INCR mode and the burst length depends on the shortest of the following factors:

- Burst length configured in the burst length field of the Watermark Level register
- Watermark level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 KB address boundary defined in the AHB protocol

The Write operation functions in a similar manner—sequential and contiguous access is necessary to ensure that the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, the byte order is swapped after the data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, the byte order is swapped before the data is stored in the buffer.

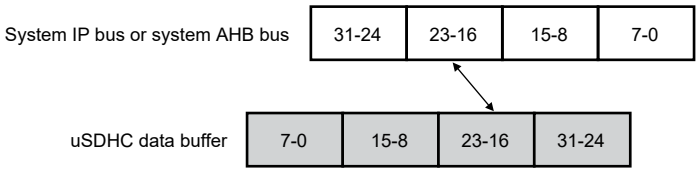


Figure 357. Data swap between system bus and uSDHC data buffer in the byte little endian mode

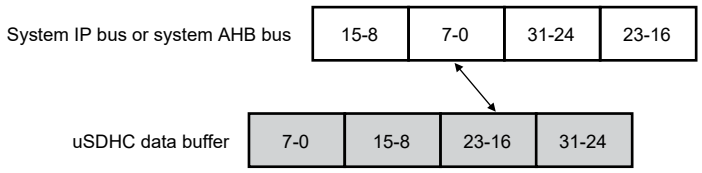


Figure 358. Data swap between system bus and uSDHC data buffer in half word big endian mode

65.3.2.1 Write operation sequence

There are 2 ways to write data into the buffer when the user transfers data to the card:

- Processor core polling through the BWR bit in the Interrupt Status register (interrupt or polling)
- Internal DMA

When the internal DMA is not used, the DMAEN bit in the Transfer Type register is not set when the command is sent, uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in WTMK_LVL[WR_WML] and is ready for receiving new data. At the same time, uSDHC sets the BWR bit. The buffer write ready interrupt is generated if it is enabled by software.

When internal DMA is used, uSDHC does not inform the system before all the required number of bytes are transferred (if no error is encountered). When an error occurs during the data transfer, uSDHC aborts the data transfer and abandons the current block. The host driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, uSDHC does not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The host driver sends CMD12 in this scenario and restarts the write operation from that address. It is recommended that a software reset for Data be applied before the transfer is restarted.

The uSDHC module does not start data transmission until the buffer has been filled with the number of words set in the WR_WML register. If the buffer is empty and the host system does not write data in time, uSDHC stops the CLK to avoid the data buffer underrun situation.

65.3.2.2 Read operation sequence

There are 2 ways to read data from the buffer when the data is received from the card:

- Processor core polling through the BRR bit in the Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, which is available and ready for system fetching data. At the same time, uSDHC sets the BRR bit. The buffer read ready interrupt is generated if it is enabled by the software.

When internal DMA is used, uSDHC does not inform the system before all the required number of bytes are transferred (if no error is encountered). When an error occurs during the data transfer, uSDHC aborts the data transfer and abandons the current block. The host driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, uSDHC does not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The host driver sends CMD12 in this scenario and restarts the read operation from that address. It is recommended that a software reset (in register RSTD) for data be applied before the transfer is restarted.

For any read transfer mode, uSDHC does not start data transmission until the number of words set in the RD_WML register are in buffer. If the buffer is full and the host system does not read data in time, uSDHC stops the CLK to avoid the data buffer overrun situation.

65.3.2.3 Data buffer and block size

In the uSDHC module, the data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly. The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way. For both read and write, the watermark level can range between 1 to 128 words. For both read and write,

the burst length can range between from 1 to 31 words. The host driver may configure the value according to the system and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes). That means, the largest block size is 512 bytes.

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned), stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the software side must write twice for each block. For each block, the ending byte is abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

65.3.2.4 Dividing large data transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in multiples of block size. If the total data length cannot be divided evenly into a multiple of the block size, then there are two options to transfer the data. These two options depend on the function and the card design. Option 1 is for the host driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data. Only for write, uSDHC sends data to the card.

See Figure 359 for an example showing the division of large data transfers, assuming a kind of WLAN SDIO that only supports a block size of up to 64 bytes. Although uSDHC supports a block size of up to 4096 bytes, SDIO can only accept a block size of less than 64 bytes, so the data must be divided (see the example below).

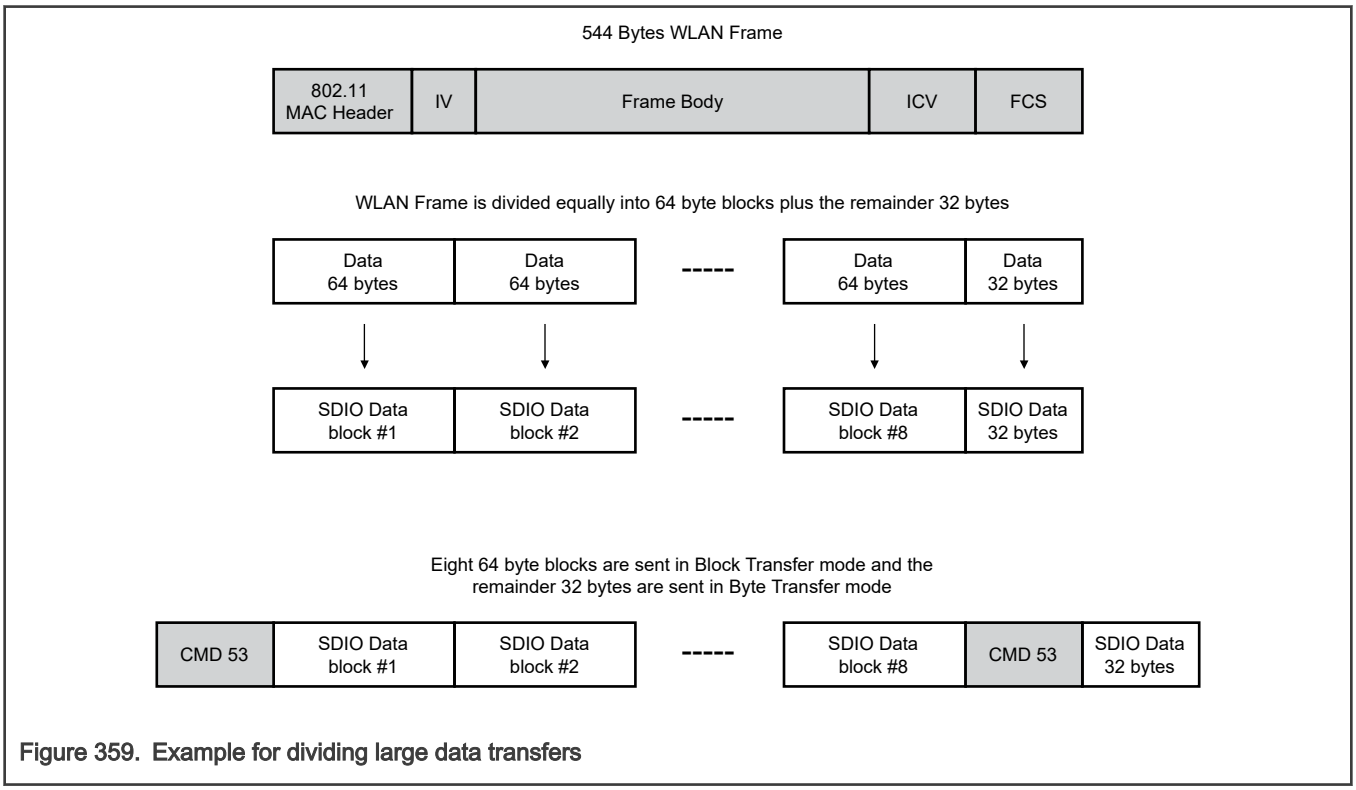


Figure 359. Example for dividing large data transfers

65.3.3 DMA AHB interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, but the BWR and BRR bits are set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register. See Figure 360 for an illustration of the DMA AHB interface block.

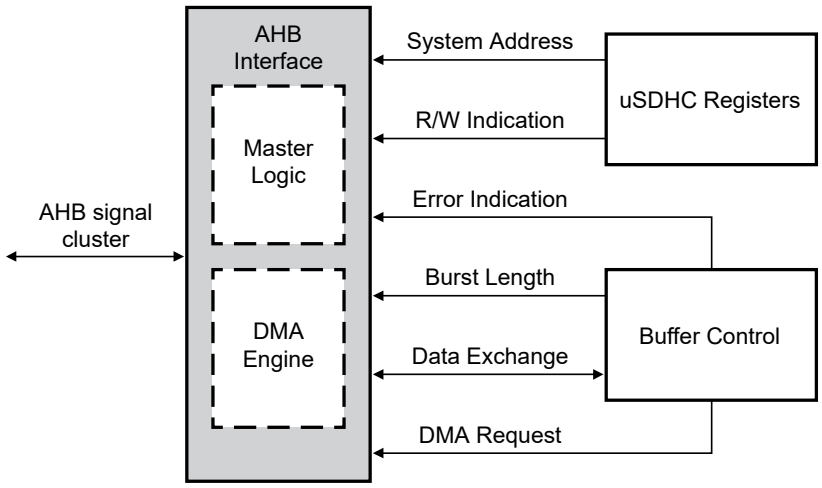


Figure 360. DMA AHB interface block

65.3.3.1 Internal DMA request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the data buffer block sends a DMA request to AHB interface.

The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The data buffer de-asserts the request if the data buffer space (for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA, the data buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of the current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always automatically set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to six words (24 bytes). After the first burst transfer, if there are greater than or equal to two words in the buffer, which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to transfer for the current block is ceiling((31 - 6 * 4) / 4) = 2. The uSDHC module reads two words out of the buffer, with seven valid bytes and one stuffed byte.

65.3.3.2 DMA burst length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can range between 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block. See the example in Internal DMA request. After six words are read, the burst length is two words, then the next burst length is six words. This is because the next block starts, which is 31 bytes, more than six words. The host driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length is the same.

65.3.3.3 AHB master interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state. At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register is generated to host CPU to report a bus error condition.

After the DMAE interrupt is received, the software sends a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and restart the transfer from this address to recover the corrupted block.

65.3.3.4 ADMA engine

In the SD host controller standard, a new DMA transfer algorithm called the Advanced DMA (ADMA) is defined. For simple DMA, after the page boundary is reached, a DMA interrupt is generated and the new system address is programmed by the host driver. The ADMA defines the programmable descriptor table in the system memory. The host driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized because the host MCU intervention is not be needed during long DMA-based data transfers.

There are two types of ADMA in host controller: ADMA1 and ADMA2. ADMA1 can support data transfer of 4KB aligned data in system memory, and ADMA2 eliminates the restriction so that the data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors defined in SD host controller Standard, and if the "End" flag is detected in the descriptor, ADMA stops after this descriptor is processed.

65.3.3.4.1 ADMA concept and descriptor format

ADMA1 includes the following descriptors:

- Valid/invalid descriptor
- Nop descriptor
- Set data length descriptor
- Set data address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

ADMA2 includes the following descriptors:

- Valid/invalid descriptor
- Nop descriptor
- Rsv descriptor (new in ADMA2)
- Set data length and address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

ADMA starts read/write operation after it reaches the tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last set type descriptor before the tran type descriptor. Every tran type descriptor triggers a transfer, and the transfer data length is extracted from the most recent set type descriptor. If there is no set type descriptor after the previous Trans descriptor, the data length is the value for previous transfer, or 0 if no set descriptor is ever met.

For ADMA2, the tran type descriptor contains both data length and transfer data address, so the tran type descriptor itself can start a data transfer.

See Figure 361 for the format of the descriptor table for ADMA1.

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 361. Format of the ADMA1 descriptor table

See Figure 362 for the concept and access method of the descriptor table for ADMA1.

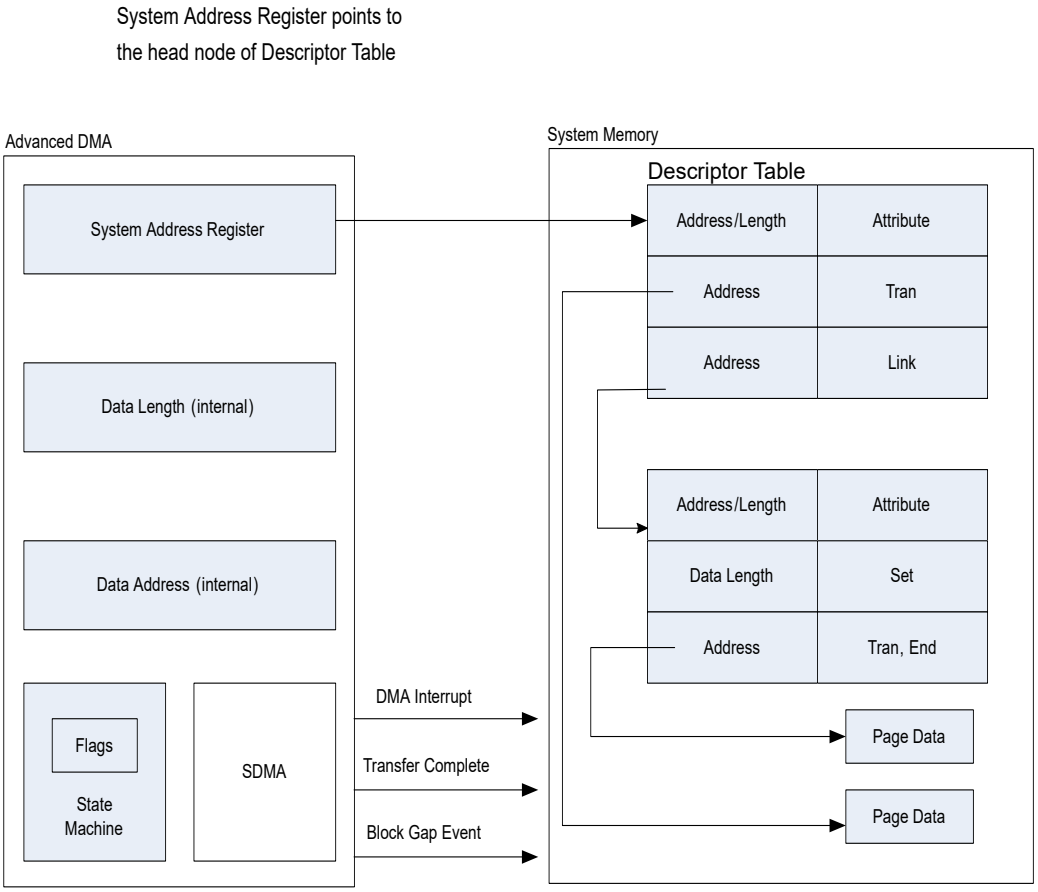


Figure 362. Concept and access method of the ADMA1 descriptor table

The figure below explains the ADMA2 format. ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it ignores the higher 32-bit. The Address field should be set to word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

Address Field		Length		Reserved		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 363. Format of the ADMA2 descriptor table

See Figure 364 for the concept and access method of the descriptor table for ADMA2.

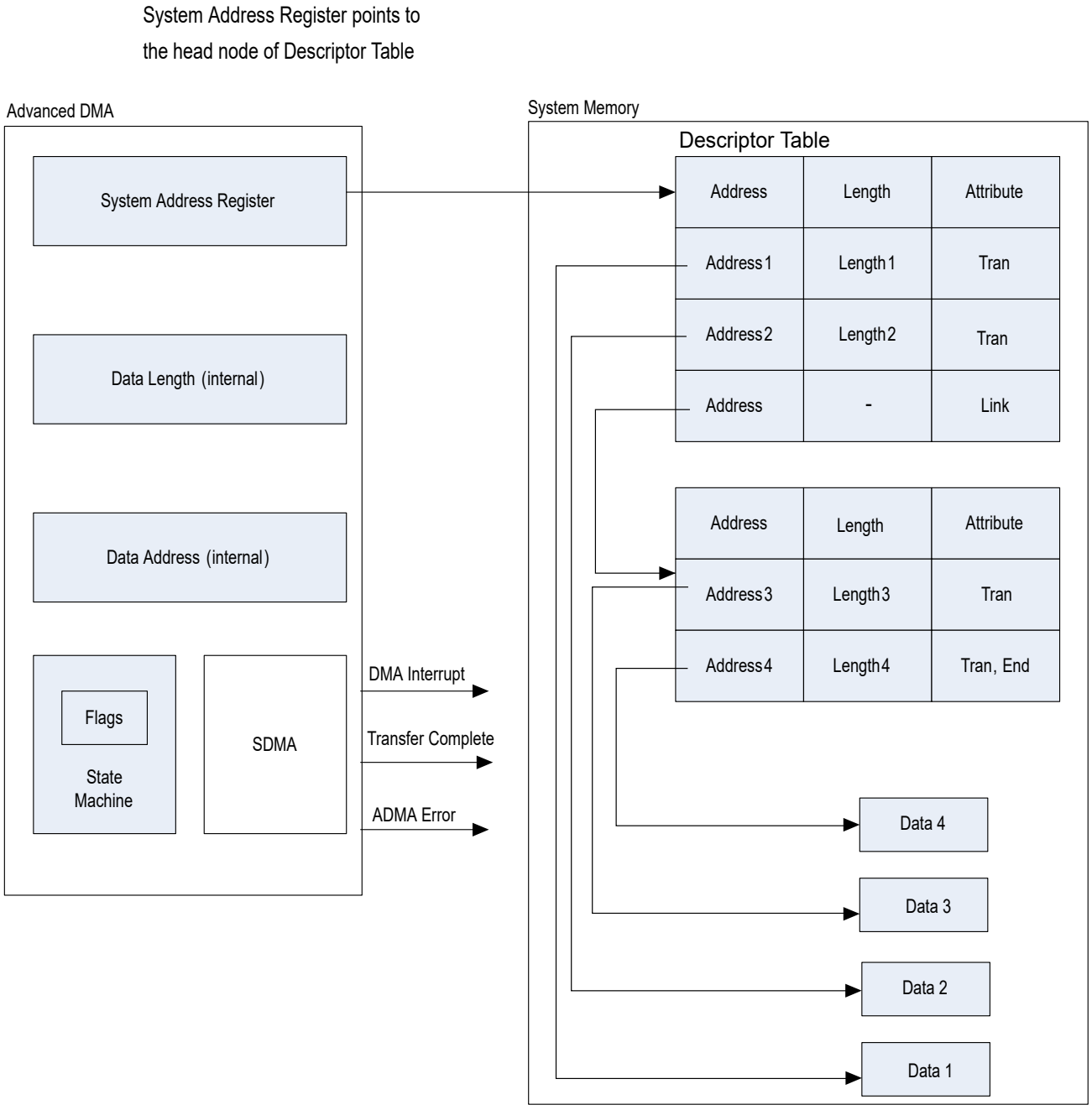


Figure 364. Concept and access method of the ADMA2 descriptor table

65.3.3.4.2 ADMA interrupt

If the interrupt flag descriptor is set, ADMA generates an interrupt according to various types of descriptors.

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.

- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

65.3.3.4.3 ADMA error

The ADMA stops whenever an error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

An ADMA descriptor error is generated when it fails to detect a "valid" flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the "Interrupt" flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in register BLK_ATT must be equal to the whole data length set in descriptor, otherwise data length mismatch error is generated.

When BLKCNTEN bit is not set, then the whole data length set in descriptor is a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors occur.

65.3.4 Register bank with IP bus interface

Register accesses through the IP bus interface are on the register bank. See Figure 365 for the block diagram.

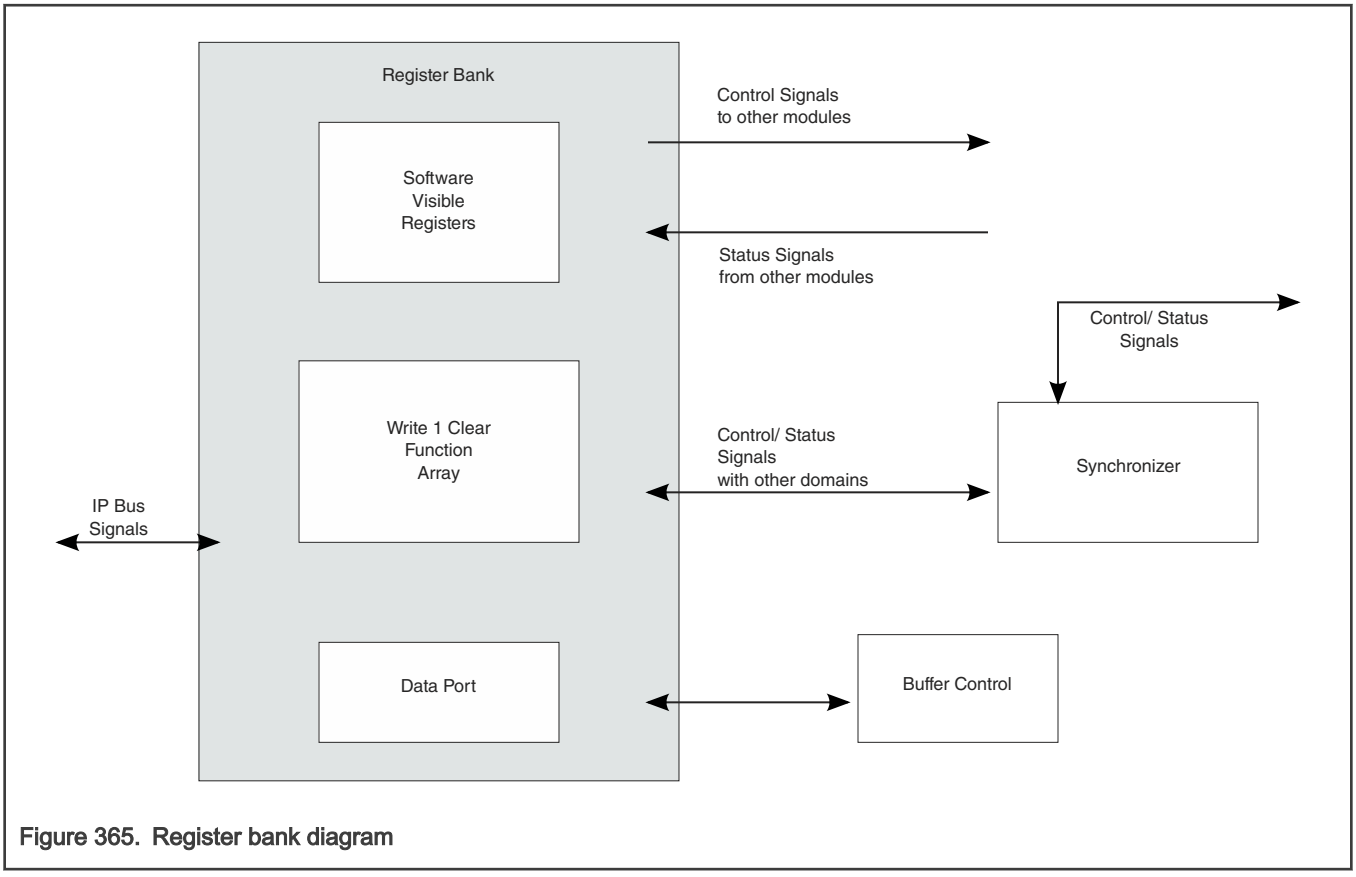


Figure 365. Register bank diagram

Only a 32-bit access is allowed, and no partial read/write is supported; therefore, all accesses are word aligned.

Access to an unimplemented address within the register memory map space does not generate a transfer error.

65.3.4.1 SD protocol unit

The SD protocol unit deals with all SD protocol affairs.

The SD protocol unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from SDIO
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status

The SD protocol unit consists of four sub-modules:

- SD control misc
- Command control
- Data control
- Clock control

65.3.4.2 SD control miscellaneous

In the SD control miscellaneous unit:

- The card detection (including DATA3 for card detection) and card interrupt are implemented.
- The module monitors the signal level on all the eight data lines and the command lines. It directly routes the level values into the register bank.

65.3.4.3 SD clock control

If the internal data buffer is near full (for read) or near empty (for write), the SD clock must be gated off to avoid buffer over/under-run. This module asserts the gate of the output SD clock to shut the clock off. After the buffer has space (for read) or has data (for write), the clock gate of this module opens, and the SD clock is active again.

65.3.4.4 Command control

The Command Control module deals with the transactions on the CMD line.

See Figure 366 for an illustration of the structure for the Command CRC shift register.

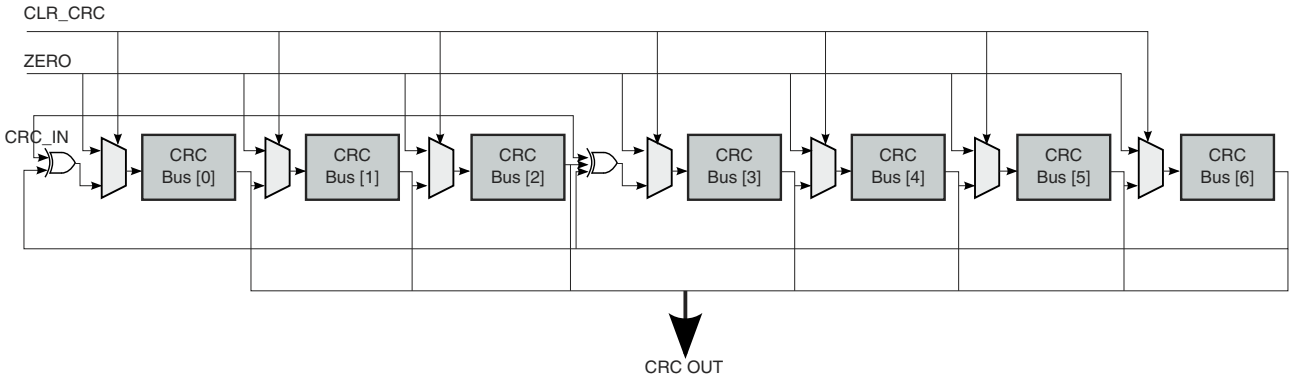


Figure 366. Command CRC shift register

The CRC polynomials for the CMD are as follows:

```
Generator polynomial: G(x) = x^7 + x^3 + 1
M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0
CRC[6:0] = Remainder [(M(x) * x^7) / G(x)]
```

65.3.4.5 Data control

The data agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line and generates the read wait state by the request from the transceiver.

The CRC polynomials for the data are as follows:

```
Generator polynomial: G(x) = x^16 + x^12 + x^5 +1

M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0

CRC[15:0] = Remainder [(M(x) * x^16) / G(x)]
```

65.3.5 Card insertion and removal detection

The uSDHC module uses the DATA3 pin to detect card insertion or removal. It is controlled by SoC pad or other logic. When there is no card on the eMMC/SD bus, the DATA3 is pulled to a low voltage level by default.

When any card is inserted to or removed from the socket, uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt.

65.3.6 Power management and wakeup events

When there is no operation between uSDHC and the card through the SD bus, the user can completely disable the peripheral clock and base clock in the chip-level clock control module to save power. When the user needs to use uSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to uSDHC are disabled, for instance, when the system is in low-power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The uSDHC module can generate these interrupts even when there are no clocks enabled. The three interrupts that can be used as wakeup events are these:

- Card removal interrupt
- Card insertion interrupt

- Interrupt from SDIO (not available in multiple block data transfers)

These three wakeup events (or wakeup interrupts) can also be used to wakeup the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to uSDHC are disabled or when the entire system is in low power mode, the corresponding wakeup enabled bit needs to be set. Refer to Protocol Control (PROT_CTRL) for more information on the uSDHC Protocol Control register.

65.3.6.1 Setting wakeup events

For uSDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters the sleep mode. Before the software disables the host clock, it should ensure that all the following conditions have been met:

- No read or write transfer is active
- Data and command lines are not active
- No interrupts are pending
- Internal data buffer is empty

65.3.7 eMMC fast boot

The Embedded MultiMediaCard (eMMC4.3 or later) specification adds a fast boot feature that requires hardware support. There are two types of fast boot modes in the eMMC4.3 or later specification: boot operation and alternative boot operation. Each type also has with-acknowledge and without-acknowledge modes.

In the boot operation mode, the master (eMultiMediaCard host) can read boot data from the slave (eMMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

NOTE

For the eMMC4.3 setting, please see the eMMC4.3 specification.

65.3.7.1 Boot operation

If the CMD line is held low for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within one second after the CMD line goes low, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line low to read all of the boot data.

NOTE

For the purposes of this documentation, fast boot is called "normal fast boot mode".

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes low. If boot acknowledge is disabled, the slave does not send out acknowledge pattern '010'.

The master can terminate the boot mode with the CMD line high.

The boot operation is terminated when all the contents of the enabled boot data are sent to the master. After the boot operation is executed, the slave gets ready for the CMD1 operation and the master needs to start a normal eMMC initialization sequence by sending CMD1.

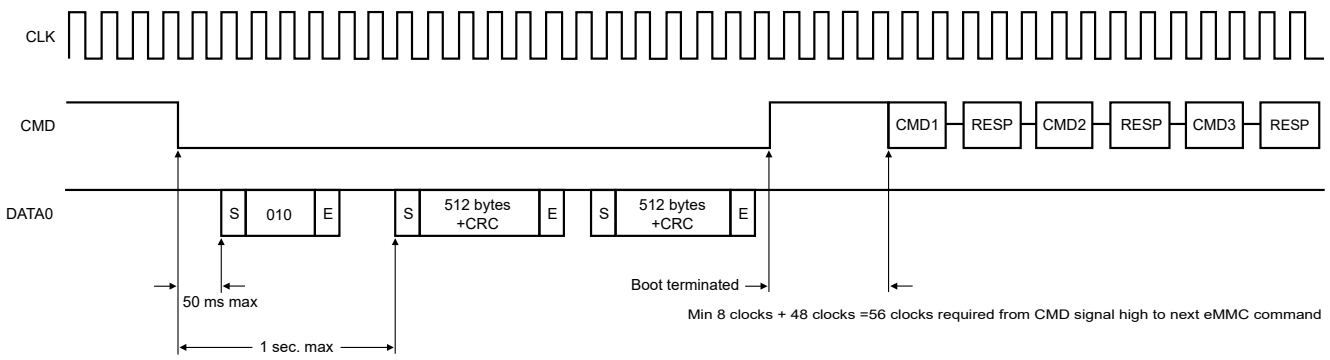


Figure 367. Embedded MultiMediaCard state diagram (normal boot mode)

65.3.7.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within one second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave must send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, the slave does not send out acknowledge pattern '010'.

The master can terminate the boot mode by issuing CMD0 (Reset).

Boot operation is terminated when all the contents of the enabled boot data are sent to the master. After the boot operation is executed, the slave gets ready for the CMD1 operation and the master needs to start a normal eMMC initialization sequence by sending CMD1.

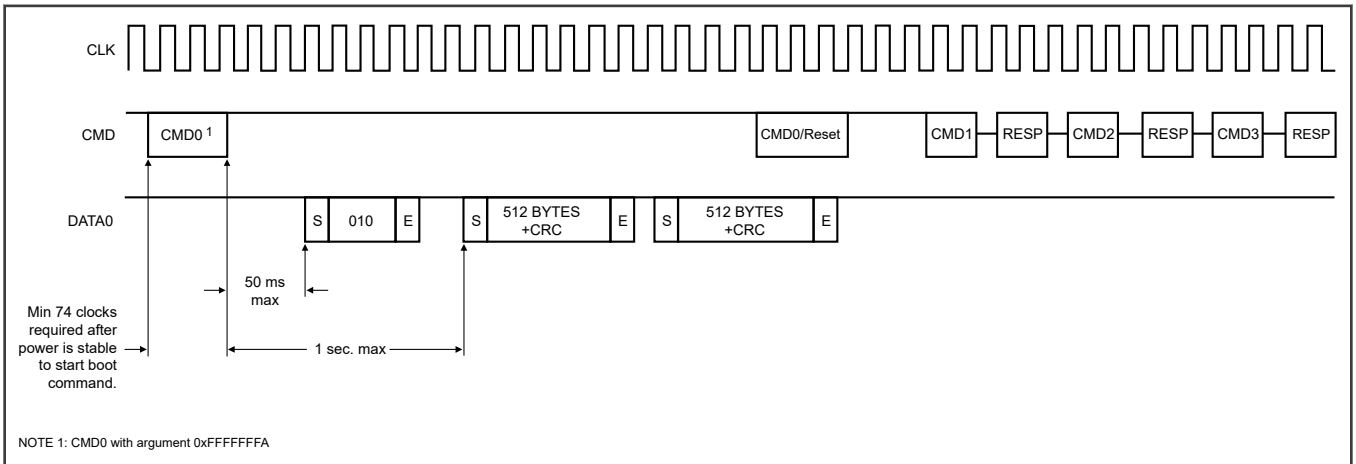


Figure 368. Embedded MultiMediaCard state diagram (alternative boot mode)

65.3.8 Commands for SD card, SDIO, and eMMC

A table containing the list of commands for the eMMC/SD card/SDIO is provided here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the SD card, SDIO, and eMMC:

- Broadcast commands (bc), no response
- Broadcast commands with response (bcr), response from all cards simultaneously
- Addressed (point-to-point) commands (ac), no data transfer on the DATA
- Addressed (point-to-point) data transfer commands (adtc)

Response: A response is a token that is sent from the card to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

Table 536. Commands for eMMC/SD card/SDIO

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all eMMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all eMMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO in idle state to send them operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The

Table continues on the next page...

Table 536. Commands for eMMC/SD card/SDIO (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
		[23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set			Embedded MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.

Table continues on the next page...

Table 536. Commands for eMMC/SD card/SDIO (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	adtc	[31:0] reserved bits(all 0)	R1	SEND_TUNING_BLOCK	64 bytes tuning pattern is sent for SDR50 and SDR104.
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21	Reserved				
CMD22	Reserved				
CMD23	ac	[31] reliable write request [30:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks (read/write) and the reliable writer parameter (write) for a block read or write command.
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command is issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write

Table continues on the next page...

Table 536. Commands for eMMC/SD card/SDIO (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
					protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command

Table continues on the next page...

Table 536. Commands for eMMC/SD card/SDIO (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
					accesses application dependent registers which are not defined in the eMMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57-59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 is used to read and write these registers.

Table continues on the next page...

Table 536. Commands for eMMC/SD card/SDIO (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62-63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default(one write block).
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on DATA3 of the card.
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for eMMC and SD cards. For eMMC, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed eMMC and high-speed SD cards. Command SWITCH_FUNC is for high-speed SD cards.
3. Command SWITCH is for high speed eMMC. The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 537.
4. ACMDs is preceded with the APP_CMD command. Commands listed are used for SD only, other SD commands not listed are not supported on this module.

The access bits for the EXT_CSD access modes are listed in the following table.

Table 537. EXT_CSD access modes

Bits	Access name	Operation
00	Command set	The command set is changed according to the Cmd Set field of the argument.
01	Set bits	The bits in the pointed byte are set, according to the bits set to 1 in the Value field.
10	Clear bits	The bits in the pointed byte are cleared, according to the bits set to 1 in the Value field.
11	Write byte	The Value field is written into the pointed byte.

65.3.9 SDIO interrupt

Information on interrupts in 1-bit mode, interrupts in 4-bit mode, and card interrupt handling are detailed in the sections below.

65.3.9.1 Interrupts in 1-bit mode

In this case, the DATA1 pin provides the interrupt function. An interrupt is asserted by pulling the DATA1 low from SDIO, until the interrupt service is finished to clear the interrupt.

65.3.9.2 Interrupt in 4-bit mode

As the interrupt and data line 1 share pin 8 in a 4-bit mode, an interrupt is only sent by the card and recognized by the host during a specific time. This is known as the interrupt period. The uSDHC module only provides samples the level on pin 8 during the interrupt period. At all other times, the host treats it as the data signal. The definition of the interrupt period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in a 4-bit mode, there is only a limited period of time that the interrupt period can be active because of the limited period of data line availability between the multiple blocks of data. This requires stricter definition of the interrupt period. For this case, the interrupt period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line holds low for one clock cycle, then pulls high for the next clock cycle . On completion of the interrupt period, the card releases the DATA1 line into the high Z state. The uSDHC module provides sample of the DATA1 during the interrupt period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Specification v1.10 for further information about the SDIO interrupt.

65.3.9.3 Card interrupt handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, uSDHC clears the interrupt request to the host system. The host driver should clear this bit before servicing the SDIO interrupt, and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from SDIO does not clear, this bit is set again. To clear this bit, it is required to reset the interrupt source from the external card followed by writing 1 to this bit. In a 1-bit mode, uSDHC detects the SDIO interrupt with or without the SD clock (to support wakeup). In a 4-bit mode, the interrupt signal is sampled during the interrupt period, so there are some sample delays between the interrupt signal from SDIO and the interrupt to the Host System Interrupt Controller. When the SDIO status is set and the host driver needs to service this interrupt, the SDIO bit in the Interrupt Control Register of SDIO is cleared. This is required to clear the SDIO interrupt status latched in uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The host driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1 and uSDHC starts sampling the interrupt signal again.

See Figure 369 for an illustration of the SDIO interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

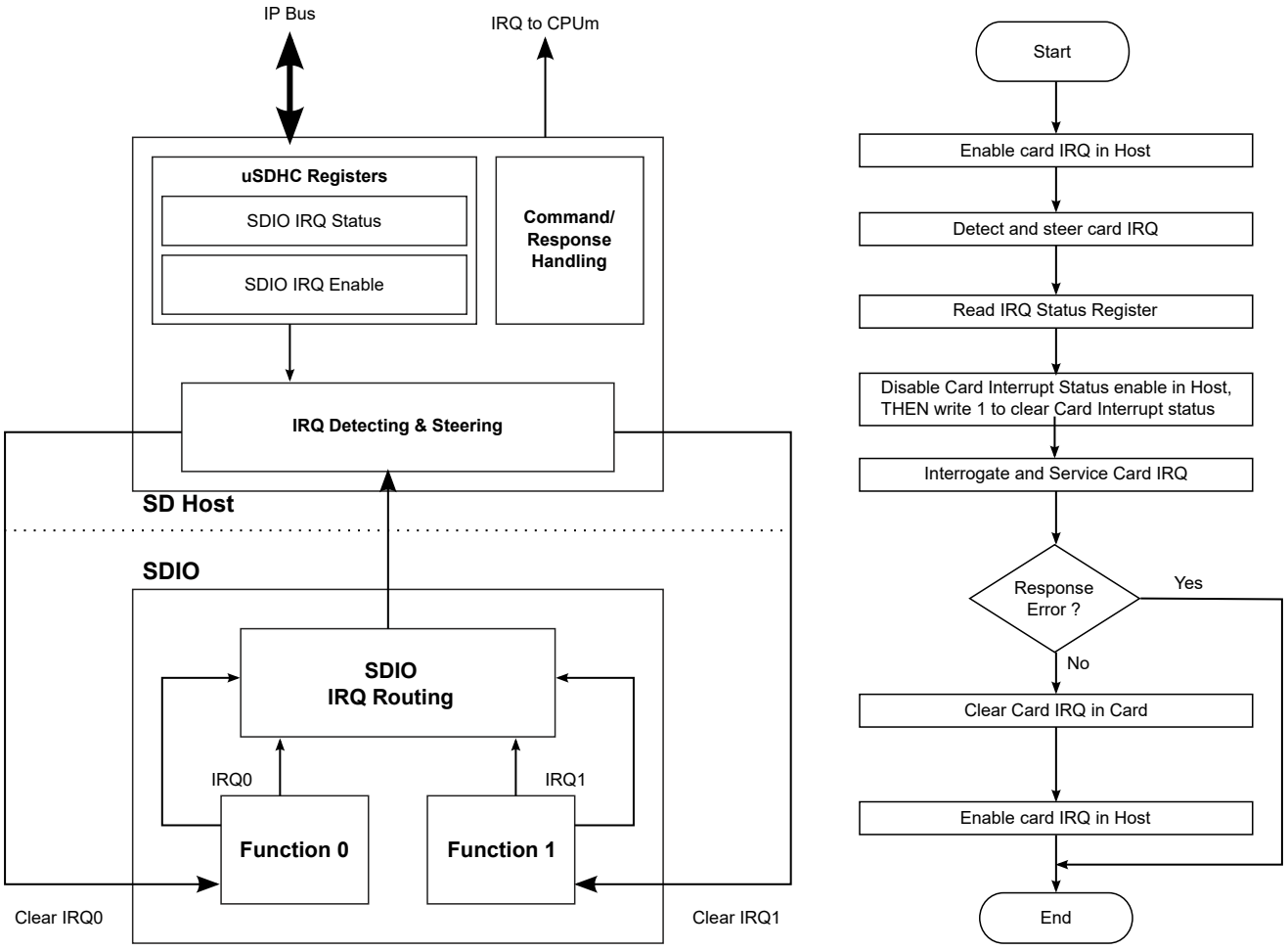


Figure 369. Card interrupt scheme, card interrupt detection, and handling procedure

65.3.10 Software restrictions

65.3.10.1 Initialization active

The driver cannot set INITA bit in System Control register when any of the command line or data lines are active, so the driver must ensure both CDIHB and CIHB bits are cleared.

65.3.10.2 Software polling procedure

For polling read or write, after the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in the Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block. For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

65.3.10.3 Suspend operation

To suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by SDIO, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended transfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such a 'suspend' command, uSDHC treats the current transfer is aborted and change the BLKCNT register to its original value, instead of retaining the remaining number of blocks.

65.3.10.4 Data length setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

ADMA1 is 4KB aligned.

65.3.10.5 (A)DMA address setting

To configure the ADMA1/ADMA2/DMA address register, when TC bit is set, the register always updates itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring the ADMA1/ADMA2/DMA address register.

65.3.10.6 Data port access

Data port does not support parallel access. For example, during an internal DMA access, it is not allowed to write any data to the data port by CPU; or during a CPU read operation, it is also prohibited to write any data to the data port, by either CPU or internal DMA. Otherwise the data is corrupted inside the uSDHC buffer.

65.3.10.7 Change clock frequency

The uSDHC module does not automatically gate off the card clock when the host driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing the clock divisor value (SDCLKFS or DVS in System Control Register) or setting the RSTA bit.

Also, before changing the clock divisor value, the host driver should make sure that the SDSTB bit is high.

65.3.10.8 Multi-block read

For pre-defined multi-block read operation, that is, the number of blocks to read has been defined by previous CMD23 for eMMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode. In this case, the card may not respond to this extra abort command and uSDHC gets response timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

65.3.11 Clocking

The table found here describes the clock sources for uSDHC. For more information on clocking, see the Clocking chapter.

Table 538. Clocks

Clock name	Description
hclk	Bus clock
ipg_clk	Peripheral clock
ipg_clk_perclk	Base clock
ipg_clk_lp	Low power clock

65.3.11.1 Clock and reset manager

This module controls all four kinds reset signals within uSDHC:

- Hardware reset
- Software reset for all logic
- Software reset for the data logic
- Software reset for the command logic

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

65.3.11.2 Clock generator

The clock generator generates the card CLK by peripheral source clock in two stages. The clock divisor can be configured through register SYS_CTRL[SDCLKFS] for prescaler configuration while the [DVS] for divisor configuration. Details can be found in the register function description. See the following figure for the structure of the divider. The term "Base" represents the frequency of peripheral source clock (see ipg_clk_perclk in Table 538).

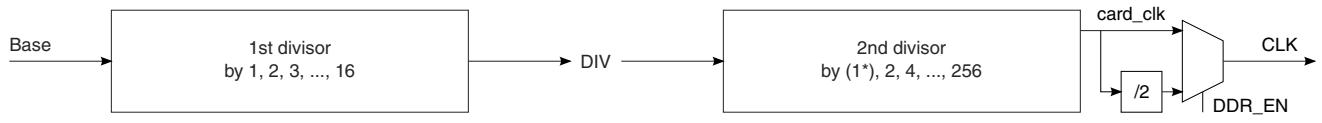


Figure 370. Two stages of the clock divider

The first stage outputs an intermediate clock (DIV) that can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler and outputs the actual internal working clock (card_clk). This clock is the driving clock for all the sub modules of the SD protocol unit, and helps in syncing FIFOs (see Figure 356) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage can be DIV, DIV/2, DIV/4,..., or DIV/256. Therefore, the highest frequency of the card_clk is base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of the base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

NOTE

CLK is different for the SDR and DDR modes.

- In the SDR mode, CLK is equal to the internal working clock (card_clk).

$$f_{card_clk} = \frac{f_{Base}}{SYS_CTRL[DVS] \times SYS_CTRL[SDCLKFS]}$$

$$f_{CLK} = f_{card_clk}$$

Equation 27. Equation for f_{card_clk} in SDR mode

$$f_{card_clk} = \frac{400MHz}{1x(2)} = 200MHz$$

$$f_{CLK} = f_{card_clk} = 200\ MHz$$

- In the DDR mode, CLK is equal to card_clk/2

$$f_{card_clk} = \frac{f_{Base}}{SYS_CTRL[DVS] \times (SYS_CTRL[SDCLKFS]/2)}$$

$$f_{CLK} = \frac{f_{card_clk}}{2}$$

Equation 28. Equation for f_{card_clk} in DDR mode

$$f_{card_clk} = \frac{400MHz}{1x(2/2)} = 400MHz$$

$$f_{CLK} = \frac{f_{card_clk}}{2} = \frac{400MHz}{2} = 200MHz$$

65.4 External signals

The following table describes the uSDHC external signals:

Table 539. uSDHC external signals

Signal	Description	Direction
CLK	Is an Internally generated clock used to drive the eMMC, SD card, and SDIO.	O
CMD	Is used to send commands and receive responses to and from the card.	I/O
DAT7	DAT7 line in the 8-bit mode — Not used in other modes	I/O
DAT6	DAT6 line in the 8-bit mode — Not used in other modes	I/O
DAT5	DAT5 line in the 8-bit mode — Not used in other modes	I/O
DAT4	DAT4 line in the 8-bit mode — Not used in other modes	I/O
NOTE If uSDHC needs to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.		
DAT3	DAT3 line in the 4/8-bit mode or configured as card detection pin. The bit may be configured as card detection pin in the 1-bit mode.	I/O
DAT2	DAT2 line or Read Wait in the 4-bit mode Read Wait in 1-bit mode	I/O
DAT1	DAT1 line in the 4/8-bit mode Also, used to detect interrupt in 1/4-bit mode	I/O
DAT0	DAT0 line in all the modes Also, used to detect busy state	I/O

65.5 Application information

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, and ALL_SEND_CID. In the Broadcast mode, eMMC are in the open-drain mode to avoid bus contention. See Commands for SD card, SDIO, and eMMC for the commands of bc and bcr categories.

After the broadcast command CMD3 is issued, the cards enter the standby mode. Addressed type commands are used from this point. In this mode, for eMMC, the CMD/DATA I/O pads turns to the push-pull mode to have the driving capability for maximum frequency operation. See Commands for SD card, SDIO, and eMMC for the commands of ac and adtc categories.

A uSDHC FIFO control problem exists with a software reset for a single block read. See MIX_CTRL[DTDSEL] for more information.

65.5.1 Command send and response receive basic operation

Assuming that the data type WORD is an unsigned 32-bit integer, the flow indicated below presents a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)

{

    WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner

    wCmd = (<cmd_index> & 0x3f) << 24; // set the first 8 bits as '00'+<cmd_index>

    set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL according to the command index;

    if (internal DMA is used) wCmd |= 0x1;

    if (multi-block transfer) {

        set MSBSEL bit;

        if (finite block number) {

            set BCEN bit;

            if (auto12 command is to use) set AC12EN bit;

        }

    }

    write_reg(CMDARG, <cmd_arg>); // configure the command argument

    write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command

}

wait_for_response(cmd_index)

{

    while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set

    read IRQ Status register and check if any error bits about Command are set
```

```
    if (any error bits are set) report error;

    write 1 to clear CC bit and all Command Error bits;

}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure that the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and move to the Standby state no response to the Host when CMD2 is sent. The host driver deals with "fake" errors like this with caution.

65.5.2 Card identification mode

When a card is inserted to the socket or the card is reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for eMMC.

65.5.2.1 Card detect

See Figure 371 for a flow diagram showing the detection of SD card and SDIO using uSDHC.

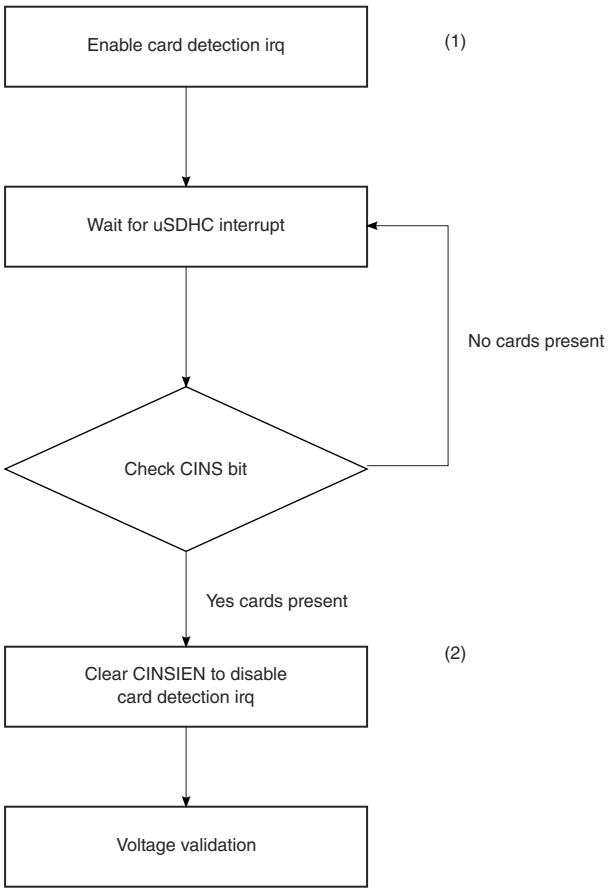


Figure 371. Flow diagram for card detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt.

- When an interrupt from uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion.
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards.

65.5.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) that is driven by Power On Reset (POR).
- Software reset (Host only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the host controller, respectively.
- Card reset (Card only): The command, "Go_Idle_State" (CMD0), is the software reset command for all types of eMMC and SD memory cards. This command sets each card into the idle state regardless of the current card state. For an SDIO, CMD52 is used to write an I/O reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both uSDHC and the card.

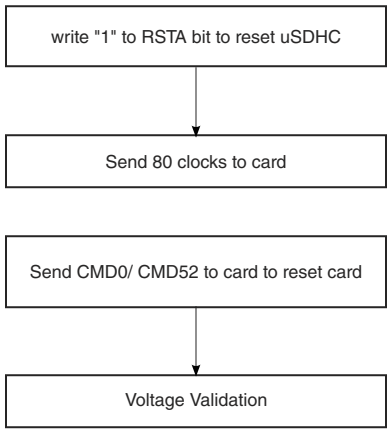


Figure 372. Flow chart for resetting uSDHC and SD card or SDIO

```
software_reset() {
    set_bit(SYSCTRL, RSTA); // software reset the Host set DTOCV and SDCLKFS fields to get the CLK of
    //frequency around 400kHz
    //configure IO pad to set the power voltage of external card to around 3.0V
    //poll bits CIHB and CDIHB
    //bits of PRSSTAT to wait both fields are cleared
    set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
    send_command(CMD_GO_IDLE_STATE, <other parameter>); // reset the card with CMD0 or
    send_command(CMD_IO_RW_DIRECT, <other parameter>);
}
```

65.5.2.3 Voltage validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for VDD are defined in the Operation Conditions Register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer VDD conditions. This means that if the host and card have non-common VDD ranges, the card is neither able to complete the identification cycle nor able to send CSD data.

Therefore, special commands Send_Op_Cont (CMD1 for eMMC), SD_Send_Op_Cont (ACMD41 for SD Memory), and IO_Send_Op_Cont (CMD5 for SD I/O) are used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards that do not match the VDD range(s) desired by the host. This is accomplished when the host sends the desired VDD voltage window as the operand of this command. Cards that cannot perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive state. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive state. This query should be used if the host is able to select a common voltage range or if a notification is sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_argument) {
    label the card as UNKNOWN;
    send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO operation
voltage, command argument is zero
    if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
        if (0 < number of IO functions) {
            label the card as SDIO;
            IORDY = 0;
            while (!(IORDY in IO OCR response)) { // set voltage range for each IO function
                send_command(IO_SEND_OP_COND, <voltage range>, <other parameter>);
                wait_for_response(IO_SEND_OP_COND);
            } // end of while ...
        } // end of if (0 < ...
        if (memory part is present inside SDIO)
            Label the card as SDCombo; //this is an SD-Combo card
    } // end of if (RESP_TIMEOUT ...
    if (the card is labeled as SDIO)
        return; // card type is identified and voltage range is set, so exit the function;
    send_command(APP_CMD, 0x0, <other parameter>); // CMD55, Application specific CMD prefix
    if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
        send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage range for
memory part or SD card
        wait_for_response(SD_APP_OP_COND); // voltage range is set
        if (card type is UNKNOWN)
           label the card as SD;
        return; //
    } // end of if (no error ...
    else if (errors other than time-out occur) { // command/response pair is corrupted
        deal with it by program specific manner;
    } // of else if (response time-out
    else { // CMD55 is refuse, it must be eMMC
        if (card is already labeled as SDCombo) { // change label
            re-label the card as SDIO;
            ignore the error or report it;
            return; // card is identified as SDIO
        } // end of if (card is ...
        send_command(SEND_OP_COND, <voltage range>, <...>);
        if(RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
            either label the card as UNKNOWN;
            return;
        } // end of if (RESP_TIMEOUT ...
    } // end of else
}
```

65.5.2.4 Card registry

This section briefly describes the registry flow. For details, please refer to the card specifications. Card registry for the eMMC and SD/SDIO/SD combo cards are different. For the SD card, the identification process starts at a clock rate lower than 400

kHz and the power voltage higher than 2.7 V (as defined by the card spec). Currently, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host requests the card to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command is sent to all the new cards in the system. Incompatible cards are put into the Inactive state. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready state), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification state.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA is used to address the card for future data transfer operations. After the RCA is received, the card changes its state to the Standby state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in the system.

For eMMC operation, the host starts the card identification process in the open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line to allow parallel card operation during card identification. After the bus is activated, the host requests the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired AND " operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive state. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in the Ready state) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. As the CID is unique for each card, only one card can be successfully sent its full CID to the host. This card then goes into the Identification state. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign a relative card address (RCA) to the card. After the RCA is received, the state of the card changes to standby, and the card does not react in further identification cycles. Also, its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

The following steps show how to perform an operation using eMMC:

```
card_registry() {
    do { // decide RCA for each card until response time-out
        if(card is labelled as SDCombo or SDIO) { // for SDIO like device
            send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO to publish its RCA
            retrieve RCA from response;
        } // end if (card is labelled as SDCombo ...
        else if (card is labelled as SD) { // for SD card
            send_command(ALL_SEND_CID, <...>);
            if(RESP_TIMEOUT == wait_for_response(ALL_SEND_CID))
                break;
            send_command(SET_RELATIVE_ADDR, <...>);
            retrieve RCA from response;
        } // else if (card is labelled as SD ...
        else if (card is labelled as eMMC) {
            send_command(ALL_SEND_CID, <...>);
            rca = 0x1; //arbitrarily set RCA, 1 here for example
            send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper 16 bits
        } // end of else if (card is labelled as eMMC...
    } while (response is not time-out);
}
```

65.5.3 Card access

Information about Block Write, Block Read, Suspense Resume, ADMA Usage, Transfer Error, and Card Interrupt are detailed in the sections below.

65.5.3.1 Block write

Information on Normal Write, DDR Write, and Write with Pause are detailed in the sections below.

65.5.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card indicates that the failure on the DATA line. The transferred data is discarded and not written, and all further transmitted blocks (in multiple block write mode) are ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write-protected area.

For eMMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO at any time, and the card responds with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby state and release the DATA line without interrupting the write operation. When re-selecting the card, it reactivates the busy indication by pulling data to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card that incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other methods (CPU polling status with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   - For eMMC and SD cards, use SET_BLOCKLEN (CMD16).
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or another error that occurred during the auto12 command sending and response receiving.

65.5.3.1.2 DDR write

uSDHC supports the dual data rate mode.

The software flow to write to a card in the DDR mode is described as below:

1. Check the card status and wait until the card is ready for data.

NOTE

For eMMC, block length only can be set to 512byte.

2. Set the uSDHC number block register (NOB), where nob is 5, for instance.
3. Set the eMMC, SD card, or SDIO to high-speed mode and use SWITCH(CMD6).
4. Set the eMMC bus or SD with 4-bit/8-bit DDR mode and use SWITCH(CMD6).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The DDR_EN and AC12EN bits should be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred or another error that occurred during the auto12 command sending and response receiving.

65.5.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the host driver, the driver can set the Stop At Block Gap Request (SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Similar to the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status and wait until the card is ready for data.
2. Set the card block length/size:
   - For eMMC and SD cards, use SET_BLOCKLEN (CMD16).
   - For SDIO or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred or some another error that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The host driver reads the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible that the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is the end of the transfer. These types of requests are ignored, the driver should treat these as a non-pause transfer, and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the host system is not stopped, and the transfer is active until the data buffer is full. Because of this, it is recommended to avoid using the Suspend command for SDIO. This is because when such a command is sent, uSDHC interprets the system that switches to another function on SDIO and flush the data buffer. uSDHC takes the Resume command as a normal command with data transfer, and it is left for the host driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC module automatically sends a CMD12 to mark the end of the multi-block transfer.

65.5.3.2 Block read

Information about Normal read, DDR read, Read with Pause, and Delay Line (DLL) in Read Path are detailed in the sections below.

65.5.3.2.1 Normal read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer state. For multi blocks read, data blocks are continuously transferred until a stop command is issued.

The software flow to read from a card that incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other methods (CPU polling status with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status and wait until card is ready for data.
2. Set the card block length/size:
   - For eMMC and SD cards, use SET_BLOCKLEN (CMD16).
   - For SDIO or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer read ready interrupt, configure the DMA settings, and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or another error occurred during the auto12 command sending and response receiving.

65.5.3.2.2 DDR read

The uSDHC module supports dual data rate mode.

The software flow to write to a card in the DDR mode is described below:

1. Check the card status and wait until the card is ready for data.

   NOTE

   For eMMC, block length can be set to only 512 bytes.

2. Set the uSDHC number block register (NOB) where nob is 5, for instance.
3. Set the eMMC, SD card, or SDIO to high-speed mode and use SWITCH (CMD6).
4. Set the eMMC bus or SD with 4-bit /8-bit DDR mode and use SWITCH(CMD6).
5. Disable the buffer write ready interrupt, configure the DMA settings, and enable the uSDHC DMA when sending the command with data transfer. The DDR_EN and AC12EN bits should be set.

6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or another error that occurred during the auto12 command sending and response receiving.

65.5.3.2.3 Read with Pause

The read operation is not generally able to pause. Only SDIO (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If SDIO supports Read Wait (SRW bit in CCCR register is 1), the host driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. Before setting the SABGREQ bit, ensure that the RWCTL bit in the Protocol Control register is set, otherwise uSDHC does not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit after the Read Wait capability of SDIO is recognized.

Similar to the flow described in Normal read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on SDIO to confirm that the card supports the Read Wait mode.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   - For eMMC and SD cards, use SET_BLOCKLEN (CMD16).
   - For SDIO or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
5. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
6. Set the uSDHC number block register (NOB), where nob is 5, for instance.
7. Disable the buffer read ready interrupt, configure the DMA setting, and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or another error occurred during the auto12 command sending and response receiving.

Similar to the Write operation, it is possible to meet the ending block of the transfer when paused. In this case, uSDHC ignores the Stop At Block Gap Request and treats it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause is transferred to the host system. No matter if the Suspend command is sent or not, the internal data buffer is not flushed.

If the Suspend command is sent and the transfer is later resumed by means of a Resume command, uSDHC takes the command as a normal one accompanied with data transfer. It is left for the host driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC automatically sends CMD12 to mark the end of multi-block transfer.

65.5.3.2.4 Delay Line (DLL) in read path

The DLL is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage, and temperature (PVT).

The reasons why DLL is needed for uSDHC are these:

- The path of read data traveling from card to host varies.
- In the eMMC and SD cards DDR mode, the minimum input setup and hold time are both at 2.5 ns.

The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in the override mode. The override value set is the number of delay cells. In the override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL automatically adjusts the number of delay cells according to the target value set by the user and PVT changes. Be aware that the target value is in units of 1/32 of the clock reference period. If the card_clk is 100Mhz, then the reference clock period is 10ns; setting target value of 16 means 5ns =(16/32)*10ns. The software can disable automatic update by the setting dll_gate_update bit.

As you might change the frequency of card_clk from time to time by changing SDCLKFS[7:0]/DVS[3:0], the software must adjust the delay value to ensure it works correctly when the reference clock (card_clk) is changed. If DLL is to be used, make sure SDCLKFS is used (at least be set to divided by 2) so that the REF_CLK generated are the same frequency as card_clk. There are two DLLs, PARTS DLL and STROBE DLL.

Step 1: Set the DLL_CTRL_RESET and DLL_CTRL_ENABLE fields

Step 2: Configure the SDCLKFS[7:0] and DVS[3:0]

Step 3: Wait until SDSTB is asserted

Step 4: Clear the DLL_CTRL_RESET field

Step 5: Wait until both the DLL_STS_SLV_LOCK and DLL_STS_REF_LOCK are asserted

Step 6: Set the DLL_CTRL_SLV_FORCE_UPD

Step 7: Clear the DLL_CTRL_SLV_FORCE_UPD

NOTE

The software should make sure that the DLL_CTRL_SLV_FORCE_UPD lasts for at least one card_clk. So, the software may need to add some delay between step 6 and step 7.

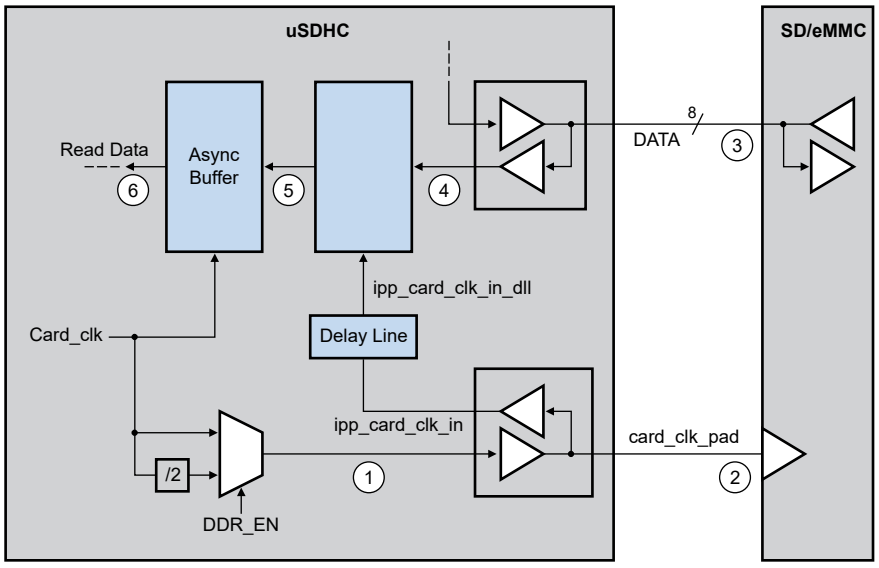


Figure 373. DLL in read path

65.5.3.3 Suspend Resume

The uSDHC module supports the Suspend Resume operations of SDIO, although slightly different than the suggested implementation of Suspend in the SDIO specification.

65.5.3.3.1 Suspend

After setting the SABGREQ bit, the host driver may send a Suspend command to switch to another function of SDIO. The uSDHC module does not monitor the content of the response, so it does not know if the Suspend command succeeded or not. Accordingly, it does not de-assert Read Wait for read pause. To solve this problem, the host driver does not mark the Suspend command as "Suspend", (that is, setting the CMDTYP bits to 01). Instead, the driver sends this command as if it were a normal command, and only when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform uSDHC that the current transfer is suspended. Here is the sequence for the Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on SDIO.

65.5.3.3.2 Resume

To resume the data transfer, a Resume command is issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation.
2. Send the Resume command: In the Transfer Type register, all the bit fields are set to the value as if this were another ordinary data transfer instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer is resumed.

65.5.3.4 ADMA usage

To use the ADMA in a data transfer, the host driver must prepare the correct descriptor chain prior to sending the read/write command. The steps to prepare the correct descriptor chain are these:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4KB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.
4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors matches the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.

7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

65.5.3.5 Transfer error

Information about CRC, Internal DMA, Transfer ADMA, and Auto CMD12 errors are detailed in the sections below.

65.5.3.5.1 CRC error

It is possible at the end of a block transfer that a write CRC status error or read CRC error occurs. For this type of error, the latest block received is discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one. For a multi-block transfer, the host driver issues a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, uSDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 is sent by uSDHC. In this case, the host driver re-sends or re-obtains the last block with a single block transfer.

65.5.3.5.2 Internal DMA error

During the data transfer with internal Simple DMA, if the DMA engine encounters an error on the AHB bus, the DMA operation is aborted, and the DMA error interrupt is sent to the host system. When acknowledged by such an interrupt, the host driver calculates the start address of data block in which the error occurs. The start address can be calculated by either:

- Reading the DMA System Address register: The error occurs during the previous burst. Considering the block size, the previous burst length and the start address of the next burst transfer, it is straight forward to obtain the start address of the corrupted block.
- Reading the BLKCNT field of the Block Attribute register: Considering the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined.To use this method, MIX_CTRL[BCEN] bit has to be set to enable Block Attribute register update.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and restart the transfer from the corrupted block to recover from the error.

65.5.3.5.3 Transfer ADMA error

There are three kinds of possible ADMA errors: The AHB transfer, invalid descriptor, and data-length mismatch. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set. For acknowledging the status, the host driver should recover the error as shown below and re-transfer from the place of interruption.

- AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or to the next block if no block is corrupted.
- Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and recreate the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
- Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

65.5.3.5.4 Auto CMD12 error

After the last block of the multi-block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, uSDHC automatically sends a CMD12 to the card to stop the transfer. When errors with this command occur, it is recommended to the host driver to deal with the situations in the following manner:

- Auto CMD12 response time-out: It is not certain whether the command is accepted by the card or not. The host driver clears the auto CMD12 error status bits and re-send CMD12 until it is accepted by the card.
- Auto CMD12 response CRC error: As the card responds to CMD12, it aborts the transfer. The host driver may ignore the error and clear the error status bit.
- Auto CMD12 conflict error or not sent: The command is not sent; therefore, the host driver sends a CMD12 manually.

65.5.3.6 Card interrupt

The external cards can inform the host controller by means of some special signals. For SDIO, it can be the low-level on the DATA1 line during some special period. The uSDHC module only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by uSDHC, and the host system is informed by uSDHC asserting the uSDHC interrupt line, the interrupt service from the host driver is called.

As the interrupt source is controlled by the external card, the interrupt from SDIO must be serviced before the CINT bit is cleared. Refer to Card interrupt handling for the card interrupt handling flow.

65.5.4 Switch function

A switch command is issued by the host driver to enable new features added to the eMMC and SD cards. The eMMC and SD cards can transfer data at bus widths other than 1-bit. Different speed modes are also defined. To enable these features, a switch command is issued by the host driver.

For SDIO, the high-speed mode /DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed as high. For SD cards, the high-speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For eMMC, the high-speed mode is queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit, DDR4-bit, and DDR8-bit width of eMMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO, it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudo-code examples do not show current capability check, which is recommended in the function switch process.

65.5.4.1 Query, enable, and disable SDIO high-speed mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report SDIO does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into uSDHC to generate the card_clk
of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is cleared;
change clock divisor value or configure the system clock feeding into uSDHC to generate the card_clk
of the desired value below 25MHz;
(data transactions like normal peers)
}
```

65.5.4.2 Query, enable, and disable SD high-speed mode/DDR50 /SDR50 /SDR104

```
enable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1 response;(high speed
mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
wait data transfer done bit is set;
check if the bit x of received 512 bits is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
      send CMD6, with argument 0x80FFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the card_clk
of around 50MHz for high speed mode, 100MHz for SDR50, 200MHz for SDR104, 50MHz for DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

65.5.4.3 Query, enable, and disable eMMC high-speed mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of eMMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the eMMC does not support high speed mode and return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of eMMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this eMMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of eMMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report eMMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of eMMC;
check if HS_TIMING byte (byte number 185) is 0;
```

```
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

65.5.4.4 Set eMMC bus width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of eMMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the eMMC does not support multiple bit width and return;
send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate), x=5;8-bit,
x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

65.5.5 ADMA operation

Here are the codes for the ADMA1 and ADMA2 operations.

65.5.5.1 ADMA1 operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB aligned);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

65.5.5.2 ADMA2 operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

65.5.6 Fast boot operation

65.5.6.1 Normal fast boot flow

Here are the steps for normal fast boot flow:

1. Software must configure the SYS_CTRL[INITA] bit to make sure that 74 card clocks are finished.
2. Software must configure the eMMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode. If the data is sent through the DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack timeout value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set Data Transfer Width (DTW). If in the DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN retain the default value, where DPSEL bit is set to 1, DTDSEL is set to 1 and MSBSEL is set to 1.
7. DMAEN should be configured as 0 in the polling mode and if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When step 6 is configured, the boot process begins. The software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot timeout happens (ack times out or the first data read times out), an interrupt is triggered, and the software must configure eMMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.

9. If there is no timeout, software needs to determine when the data read is finished and then configure eMMC Boot Register bit 6 to 0 to disable boot. This render CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin the normal initialization process.
10. You must reset the host and begin the normal process.

65.5.6.2 Alternative fast boot flow

Here are the steps for alternative fast boot flow:

1. Software needs to configure SYS_CTRL[INITA] to make sure 74 card clocks are finished.
2. Software needs to configure eMMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through the DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in the DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the data transfer width (DTW). If in the DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with the 0xFFFFFFFA argument. In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN, and DMAEN retain the default value. DPSEL bit is set to 1, DTDSEL is set to 1, and MSBSEL is set to 1. Note DMAEN should be configured as 0 in the polling mode, and if BCEN is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in the DDR fast boot mode, DDR_EN needs to be set to 1.
7. When step 6 is configured, the boot process begins. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot timeout (ack data timeout in 50ms or data timeout in 1s), the host sends out the interrupt and the software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process.
8. If there is no time out, the software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the eMMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. You must reset the host and begin the normal process.

65.5.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during eMMC fast boot.

In fast boot, the host can use Advanced DMA2 (ADMA2) with two destinations.

The detailed flow is described below:

1. The software needs to configure INIT_ACTIVE bit (system control register bit 27) to make sure that 74 card clocks are finished.
2. The software needs to configure the eMMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (BLK_CNT - VALUE1). Here VALUE1 is the value of the block count that needs to transfer the first time, so that the host stops at the block gap when the uSDHC controller gets VAULE1 blocks from the device. Also, configure bits[3-0] to select the ack timeout value according to the SD clock frequency.

3. The software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (BLK_CNT) to the max value (16'hffff).
4. The software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software enable ADMA2 by configuring Protocol Control Register bits [9-8].
6. The software needs to set at least three pairs of ADMA2 descriptor in boot memory (that is, in IRAM, at least six words). The first pair descriptor defines the start address (that is, IRAM) and data length (that is,512byte*VALUE1) of the first part boot code. The software also needs to set the second pair descriptor, the second start address (any value that is writable), and data length is suggested to set 1~2word (record as VALUE2). Note that the second couple desc also transfers useful data even at lease 1 word, because our ADMA2 cannot support 0 data_length data transfer descriptor.
7. The software needs to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot and do not need to be set in normal fast boot.
8. The software needs to configure Transfer Type Register to start the boot process. CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN, and DMAEN retain the default value. DPSEL bit is set to 1, DTDSEL is set to 1, and MSBSEL is set to 1. DMAEN is configured as 1 in the DMA mode. And, if BCEN is configured as 1, then configure blk no in Bock Attributes Register to the max value. And, if in the DDR fast boot mode, DDR_EN needs to be set to 1.
9. When step 8 is configured, boot process begins, the first VALUE1 block number data gets transferred. The software needs to poll the TC bit (bit1 in Interrupt Status Register) to determine first transfer is ended. Also, the software needs to polling the BGE bit (bit2 in Interrupt Status Register) to determine if the first transfer stops at the block gap.
10. When TC and BGE bits are set to 1, the software can analyze the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of the boot code (VALUE3, the remain boot code block). Remember to set the last descriptor with END.
11. The software needs to configure the eMMC Boot Register (offset 0xc4) again. Set bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot), to 1 (alternative boot); and bit 4 to select the ack mode or not. In the DMA mode, configure bit 7 to 1 for enabling the automatically stop at block gap feature. Also, configure bit31-bit16 to set the (BLK_CNT - (VALUE1+1+VALUE3)), that host stops at block gap when the uSDHC controller gets (VALUE1+1+VALUE3)) blocks from device totally include the blocks received in step 9. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequency. Note that the software does not need to configure the BLK_CNT again, because it is counted down automatically by the uSDHC controller.
12. The software needs to clear the TC and BGE bits and the software needs to clear SABGREQ (bit 16 in the Protocol control register) and set CREQ (bit17 in the Protocol control register) to 1 to resume the data transfer. Host transfers the VALUE2 and VALUE3 data to the destination that is set by descriptor.
13. The software needs to do poll BGE bit to determine if the fast boot is over.

Note:

- When ADMA boot flow starts, for uSDHC, it is like a normal ADMA read operation. So, set ADMA2 descriptor as the normal ADMA2 transfer.
- Need a few words length memory to keep descriptor.
- For the 1~2-word data in second descriptor setting, it is a useful data, so the software needs to deal the data because of the application case.

65.6 uSDHC memory map and register definition

65.6.1 uSDHC register descriptions

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map of uSDHC. All these registers only support 32-bit accesses.

NOTE

The uSDHC registers are 32-bit wide and only support 32-bit access.

65.6.1.1 uSDHC memory map

uSDHC0 base address: 4010_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	DMA System Address (DS_ADDR)	32	RW	0000_0000h
4h	Block Attributes (BLK_ATT)	32	RW	0001_0000h
8h	Command Argument (CMD_ARG)	32	RW	0000_0000h
Ch	Command Transfer Type (CMD_XFR_TYP)	32	RW	0000_0000h
10h	Command Response0 (CMD_RSP0)	32	R	0000_0000h
14h	Command Response1 (CMD_RSP1)	32	R	0000_0000h
18h	Command Response2 (CMD_RSP2)	32	R	0000_0000h
1Ch	Command Response3 (CMD_RSP3)	32	R	0000_0000h
20h	Data Buffer Access Port (DATA_BUFF_ACC_PORT)	32	RW	0000_0000h
24h	Present State (PRES_STATE)	32	R	See section
28h	Protocol Control (PROT_CTRL)	32	RW	0880_0020h
2Ch	System Control (SYS_CTRL)	32	RW	0080_800Fh
30h	Interrupt Status (INT_STATUS)	32	RW	0000_0000h
34h	Interrupt Status Enable (INT_STATUS_EN)	32	RW	0000_0000h
38h	Interrupt Signal Enable (INT_SIGNAL_EN)	32	RW	0000_0000h
3Ch	Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)	32	RW	0000_0000h
40h	Host Controller Capabilities (HOST_CTRL_CAP)	32	RW	07F3_B407h
44h	Watermark Level (WTMK_LVL)	32	RW	0810_0810h
48h	Mixer Control (MIX_CTRL)	32	RW	8000_0000h
50h	Force Event (FORCE_EVENT)	32	RW	0000_0000h
54h	ADMA Error Status (ADMA_ERR_STATUS)	32	R	0000_0000h
58h	ADMA System Address (ADMA_SYS_ADDR)	32	RW	0000_0000h
60h	DLL (Delay Line) Control (DLL_CTRL)	32	RW	0000_0000h
64h	DLL Status (DLL_STATUS)	32	R	0000_0200h
68h	CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)	32	RW	0000_0000h
C0h	Vendor Specific Register (VEND_SPEC)	32	RW	3000_7809h
C4h	eMMC Boot (MMC_BOOT)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
C8h	Vendor Specific 2 Register (VEND_SPEC2)	32	RW	0001_9006h
CCh	Tuning Control (TUNING_CTRL)	32	RW	0021_2800h

65.6.1.2 DMA System Address (DS_ADDR)

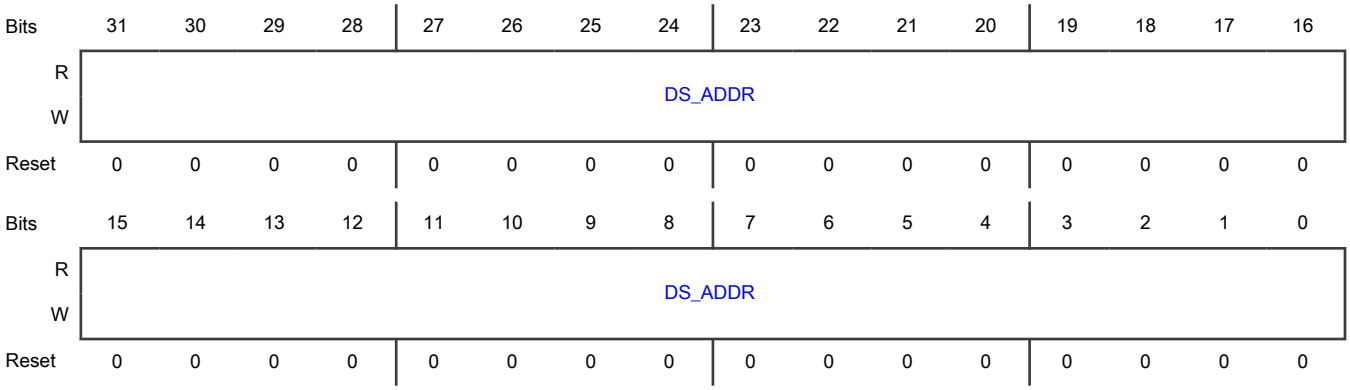
Offset

Register	Offset
DS_ADDR	0h

Function

This register contains the physical system memory address used for DMA transfers.

Diagram



Fields

Field	Function
31-0 DS_ADDR	System address **DMA system address / argument 2** When ACMD23_ARGU2_EN is set to 0, SDMA uses this register as system address and supports only 32-bit addressing mode. Auto CMD23 cannot be used with SDMA. When ACMD23_ARGU2_EN is set to 1, SDMA uses ADMA System Address register (05Fh – 058h) instead of this register to support both ADMA1 and ADMA2. This register is used only for Argument2 and SDMA may use Auto CMD23. 1. **SDMA system address**

Table continues on the next page...

Field	Function
	Because the address must be word (4 bytes) aligned, the least two bits are reserved and always set to 0. When uSDHC stops a DMA transfer, this register points out the system address of the next contiguous data position. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operation during transfers may return an invalid value. The host driver initializes this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The host driver waits until the DLA field in the Present State register is cleared, before writing to this register. The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC automatically changes SEQ burst type to NSEQ. Because this register supports dynamic address reflecting, when TC field is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC field is set. Such restriction is also listed in Software restrictions. 2. **Argument 2** This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23. If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case.

65.6.1.3 Block Attributes (BLK_ATT)

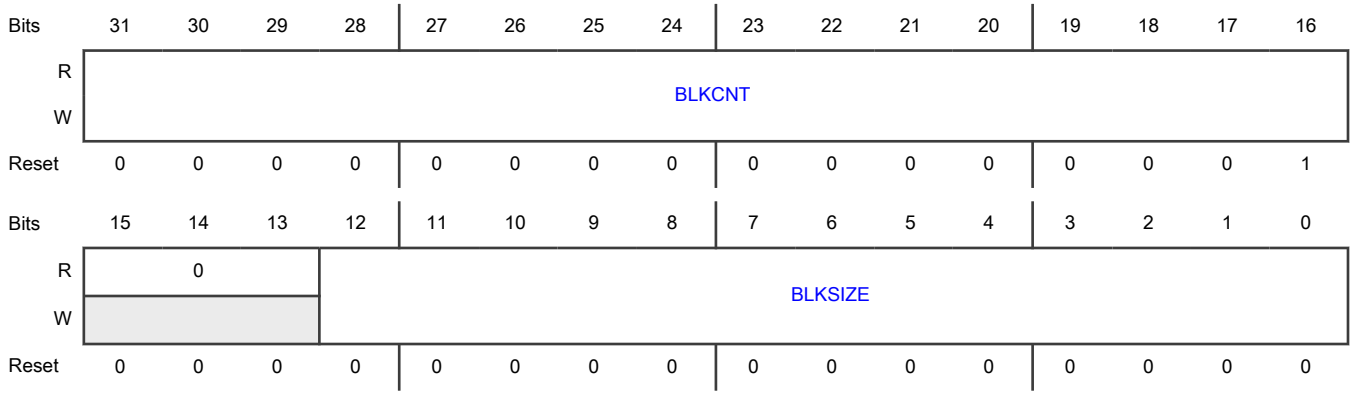
Offset

Register	Offset
BLK_ATT	4h

Function

This register is used to configure the number of data blocks and the number of bytes in each block.

Diagram



Fields

Field	Function
31-16 BLKCNT	Blocks count for current transfer This field is enabled when the Block Count Enable field in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this field always reads as 1. The host driver sets this field to a value between 1 and the maximum block count. The uSDHC module decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to zero results in no data blocks being transferred. This field should be accessed only when no transaction is executing (that is, after transactions are stopped). During data transfer, read operations on this field may return an invalid value and write operations are ignored. When saving transfer content because of a Suspend command, the number of blocks yet to be transferred can be determined by reading this field. The reading of this field should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when the Suspend command is sent out, uSDHC treats the current transfer as aborted and change the BLKCNT field back to its original value instead of keeping the dynamical indicator of the remaining block count. When restoring transfer content prior to issuing a Resume command, the host driver restores the previously saved block count. **NOTE** Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL field is indicating a single block transfer, the read value of BLKCNT is always 1. 0000_0000_0000_0000b - Stop count 0000_0000_0000_0001b - 1 block 0000_0000_0000_0010b - 2 blocks 1111_1111_1111_1111b - 65535 blocks
15-13 —	Reserved
12-0 BLKSIZE	Transfer block size This field specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations are ignored. 0_0000_0000_0000b - No data transfer 0_0000_0000_0001b - 1 byte 0_0000_0000_0010b - 2 bytes 0_0000_0000_0011b - 3 bytes 0_0000_0000_0100b - 4 bytes 0_0001_1111_1111b - 511 bytes

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0010_0000_0000b - 512 bytes 0_1000_0000_0000b - 2048 bytes 1_0000_0000_0000b - 4096 bytes

65.6.1.4 Command Argument (CMD_ARG)

Offset

Register	Offset
CMD_ARG	8h

Function

This register contains the SD/eMMC command argument.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CMDARG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMDARG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CMDARG	Command argument The SD/eMMC command argument is specified as bits 39-8 of the command format in the SD or eMMC specification. This field is write protected when the Command Inhibit (CMD) field in the Present State register is set.

65.6.1.5 Command Transfer Type (CMD_XFR_TYP)

Offset

Register	Offset
CMD_XFR_TYP	Ch

Function

This register is used to control the operation of data transfers. The host driver sets this register before issuing a command followed by a data transfer or before issuing a Resume command. To prevent data loss, uSDHC prevents writing to the bits, which are involved in the data transfer of this register, when data transfer is active. These fields are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

The host driver checks the Command Inhibit DAT field (PRES_STATE[CDIHB]) and the Command Inhibit CMD field (PRES_STATE[CIHB]) in the Present State register before writing to this register. When the CDIHB field in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB field is set, any write to this register is ignored.

On sending commands with data transfer involved, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as a single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC ignores the sending of this command and do nothing.

If the commands with data transfer do not receive the response in 64 clock cycles, that is, if response time-out happens, uSDHC treats the external device, does not accept the command, and aborts the data transfer. In this scenario, the driver should issue the command again to retry the transfer. It is also possible that for some reason the card responds to the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 540. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, regarding the Response Type bits as well as the name of the response type.

Table 541. Relationship between parameters and the name of the response type

Response type	Index check enable	CRC check enable	Name of response type
00	0	0	No response
01	0	1	R2
10	0	0	R3, R4

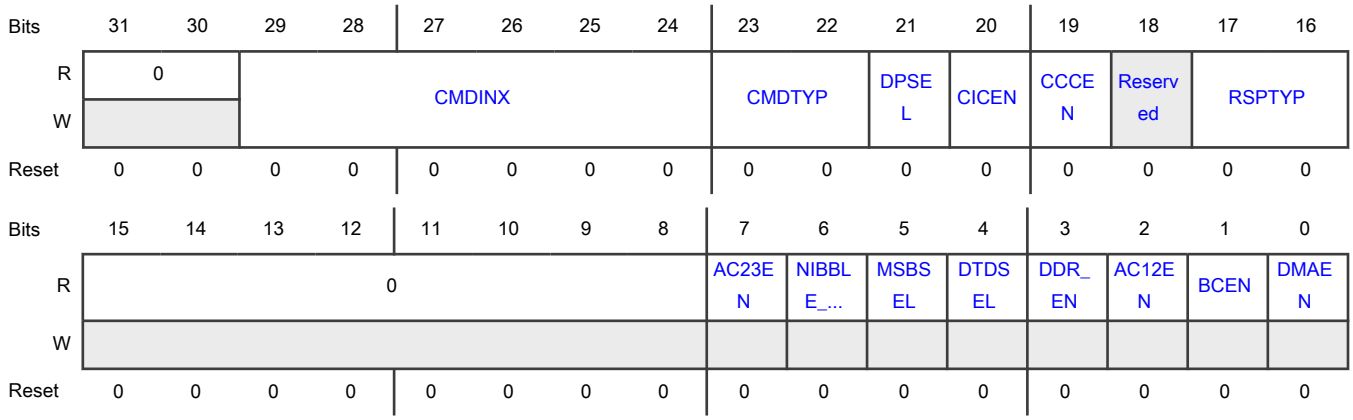
Table continues on the next page...

Table 541. Relationship between parameters and the name of the response type (continued)

Response type	Index check enable	CRC check enable	Name of response type
10	1	1	R1, R5, R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification, but R5b is defined in this specification to specify that uSDHC checks the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command is used with R5b.
- The CRC fields for R3 and R4 are expected to be all 1 bits. The CRC check is disabled for these response types.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-24 CMDINX	Command index These fields are set to the command number that is specified in bits 45-40 of the command-format in the SD Memory Card Physical Layer Specification and SDIO Specification.
23-22 CMDTYP	Command type There are three types of special commands: Suspend, Resume, and Abort. These fields are set to 00b for all other commands. • Suspend command: If the Suspend command succeeds, uSDHC assumes that the card bus has been released and that it is possible to issue the next command that uses the DATA line. Because uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the host driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform uSDHC that a Suspend command was successfully issued. See Suspend Resume for more details. After the end bit of command is sent, uSDHC deasserts Read Wait for read transactions and stops checking busy for

Table continues on the next page...

Table continued from the previous page...

Field	Function
	write transactions. In a 4-bit mode, the interrupt cycle starts. If the Suspend command fails, uSDHC maintains its current state, and the host driver restarts the transfer by setting the Continue Request field in the Protocol Control register. • Resume command: The host driver re-starts the data transfer by restoring the registers saved before sending the Suspend command and then sends the Resume command. The uSDHC module checks for a pending busy state before starting write transfers. • Abort command: If this command is set when executing a read transfer, uSDHC stops reads to the buffer. If this command is set when executing a write transfer, uSDHC stops driving the DATA line. After issuing the Abort command, the host driver should issue a software reset (Abort Transaction). 00b - Normal other commands 01b - Suspend CMD52 for writing bus suspend in CCCR 10b - Resume CMD52 for writing function select in CCCR 11b - Abort CMD12, CMD52 for writing I/O Abort in CCCR
21 DPSEL	Data present select This field is set to 1 to indicate that data is present and is transferred using the DATA line. It is set to 0 for the following: • Commands using only the CMD line (for example, CMD52) • Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b (for example, CMD38)) **NOTE** In resume command, this field is set, and other bits in this register is set the same as when the transfer was initially launched. When this field is set, while the DTDSEL field is 0, writes to the register Transfer Type are ignored. 0b - No data present 1b - Data present
20 CICEN	Command index check enable If this field is set to 1, uSDHC checks the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this field is set to 0, the Index field is not checked. 0b - Disable command index check 1b - Enables command index check
19 CCCEN	Command CRC check enable If this field is set to 1, uSDHC checks the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this field is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. See RSPTYP[1:0] and Command Transfer Type (CMD_XFR_TYP).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disables command CRC check 1b - Enables command CRC check
18 —	Reserved
17-16 RSPTYP	Response type select 00b - No response 01b - Response length 136 10b - Response length 48 11b - Response length 48, check busy after response
15-8 —	Reserved
7 AC23EN	AC23EN This field is read when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is tied to '0'. When this field is set to 1, the host controller issues a CMD23 automatically before issuing a command specified in the Command Register. 0b - Disable 1b - Enable
6 NIBBLE_POS	NIBBLE_POS This field indicates the nibble position in the DDR 4-bit mode. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is read-only. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'. 0b - Disable 1b - Enable
5 MSBSEL	MSBSEL This field enables multiple block DATA line data transfers. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is read-only. For any other commands, this field can be set to 0. If this field is 0, it is not necessary to set the Block Count register. See Command Transfer Type (CMD_XFR_TYP). 0b - Disable 1b - Enable
4 DTDSEL	DTDSEL This field defines the direction of DATA line data transfers. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is read-only. The field is set to 1 by the host driver to transfer data from the SD card to uSDHC and is set to 0 for all other commands.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If a CMD with the response data size cannot be divided by 64 bytes, is given without a STOP CMD, and is followed by a software reset, then you must write a 0 to DTDSEL before inserting a software reset. This is necessary to avoid any issue during the SD card reinitialization. Read multiple blocks will not be affected as it is required to have a STOP CMD. 0b - Disable 1b - Enable
3 DDR_EN	DDR_EN Dual data rate mode selection. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is read-only. 0b - Disable 1b - Enable
2 AC12EN	AC12EN Multiple block transfers for memory require a CMD12 to stop the transaction. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is read-only. When this field is set to 1, uSDHC issues a CMD12 automatically when the last block transfer has completed. The host driver is not set this field to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, uSDHC ignores this field no matter it is set or not. 0b - Disable 1b - Enable
1 BCEN	BCEN This field is used to enable the Block Count register, which is only relevant for multiple block transfers. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is read-only. When this field is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 0b - Disable 1b - Enable
0 DMAEN	DMAEN This field enables DMA functionality. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is enabled; otherwise, this field is read-only. If this field is set to 1, a DMA operation begins when the host driver sets the DPSEL field of this register. Whether the simple DMA or the advanced DMA is active depends on the DMA Select field of the Protocol Control register. 0b - Disable 1b - Enable

65.6.1.6 Command Response0 (CMD_RSP0)

Offset

Register	Offset
CMD_RSP0	10h

Function

This register is used to store part 0 of the response bits from the card.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CMDRSP0	Command response 0 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

65.6.1.7 Command Response1 (CMD_RSP1)

Offset

Register	Offset
CMD_RSP1	14h

Function

This register is used to store part 1 of the response bits from the card.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CMDRSP1	Command response 1 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

65.6.1.8 Command Response2 (CMD_RSP2)

Offset

Register	Offset
CMD_RSP2	18h

Function

This register is used to store part 2 of the response bits from the card.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CMDRSP2	Command response 2 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

65.6.1.9 Command Response3 (CMD_RSP3)

Offset

Register	Offset
CMD_RSP3	1Ch

Function

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD bus to Command Response registers for each response type. In this table, R[ ] refers to a bit range within the response data as transmitted on the SD bus.

Table 542. Response bit definition for each response type

Response type	Meaning of response	Response field	Response register
R1,R1b (normal response)	Card status	R[39:8]	CMDRSP0
R1b (auto CMD12 response)	Card status for auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (publish RCA)	New published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, uSDHC only stores part of the response data in the Command Response registers. This enables the host driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by uSDHC (as specified by the Command Index Check Enable and the

Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, uSDHC checks R[47:1], and if the response length is 136 the uSDHC checks R[119:1].

Because uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CMDRSP3	Command response 3 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

65.6.1.10 Data Buffer Access Port (DATA_BUFF_ACC_PORT)

Offset

Register	Offset
DATA_BUFF_ACC_PORT	20h

Function

The Buffer Data Port register is for 32-bit data access by the Arm platform. When the internal DMA is enabled, any write to this field is ignored, and any read from this field always yields 0s.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATCONT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATCONT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DATCONT	Data content This field is used to access the internal buffer.

65.6.1.11 Present State (PRES_STATE)

Offset

Register	Offset
PRES_STATE	24h

Function

The host driver can get status of uSDHC from this 32-bit read only register.

The host driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands are issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

NOTE

The reset value of Present State register depends on board connectivity.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLSL								CLSL	0			Reserved	Reserved	0	CINST
W																
Reset	u	u	u	u	u	u	u	u	u	0	0	0	u	u	0	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TSCD	0		RTR	BREN	BWEN	RTA	WTA	Reserved				SDSTB	DLA	CDIHB	CIHB
W																
Reset	1	0	0	0	0	0	0	0	1	0	0	0	u	0	0	0

Fields

Field	Function
31-24 DLSL	DATA[7:0] line signal level This field is used to check the DATA line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up. 0000_0001b - Data 0 line signal level 0000_0010b - Data 1 line signal level 0000_0100b - Data 2 line signal level 0000_1000b - Data 3 line signal level 0001_0000b - Data 4 line signal level 0010_0000b - Data 5 line signal level 0100_0000b - Data 6 line signal level 1000_0000b - Data 7 line signal level
23 CLSL	CMD line signal level This field is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this field after reset is 1'b1, when the command line is pulled up.
22-20 —	Reserved
19 —	Reserved
18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 —	Reserved
16 CINST	Card inserted This field indicates whether a card has been inserted. The uSDHC module debounces this signal so that the host driver does not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not affect this field. The Software Reset For All in the System Control register does not affect this field. A software reset does not affect this field. 0b - Power on reset or no card 1b - Card inserted
15 TSCD	Tap select change done This field indicates the delay setting is effective after write CLK_TUNE_CTRL_STATUS register. 0b - Delay cell select change is not finished. 1b - Delay cell select change is finished.
14-13 —	Reserved
12 RTR	Re-Tuning Request (only for SD3.0 SDR104 mode) Host controller may request host driver to execute re-tuning sequence by setting this field when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This field is cleared when a command is issued with setting Execute Tuning field in MIX_CTRL register. Changing of this field from 0 to 1 generates Re-Tuning Event. See Interrupt status registers for more detail. This field isn't set to 1 if Sampling Clock Select in the MIX_CTRL register is set to 0 (using fixed sampling clock). 0b - Fixed or well tuned sampling clock 1b - Sampling clock needs re-tuning
11 BREN	Buffer read enable This status field is used for non-DMA read transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this field is high, valid data greater than the watermark level exist in the buffer. A change of this field from 1 to 0 occurs when some reads from the buffer (read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this field from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 0b - Read disable 1b - Read enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 BWEN	Buffer write enable This status field is used for non-DMA write transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this field is 1, valid data greater than the watermark level can be written to the buffer. A change of this field from 1 to 0 occurs when some writes to the buffer (write DATPORT (Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this field from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 0b - Write disable 1b - Write enable
9 RTA	Read transfer active This status field is used for detecting completion of a read transfer. This field is set for either of the following conditions: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer A transfer complete interrupt is generated when this field changes to 0. This field is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, that is, all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent because of the Stop At Block Gap Request being set to 1. 0b - No valid data 1b - Transferring data
8 WTA	Write transfer active This status field indicates a write transfer is active. If this field is 0, it means no valid write data exists in uSDHC. This field is set in either of the following cases: • After the end field of the write command • When writing 1 to the Continue Request field in the Protocol Control register to restart a write transfer This field is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple) • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request During a write transaction, a Block Gap Event interrupt is generated when this field is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the host driver in determining when to issue commands during Write Busy state.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No valid data 1b - Transferring data
7-4 —	Reserved
3 SDSTB	SD clock stable This status field indicates that the internal card clock is stable. This field is for the host driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON field in System Control register to remove glitches on the card clock when the frequency is changing. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. 0b - Clock is changing frequency and not stable. 1b - Clock is stable.
2 DLA	Data line active This status field indicates whether one of the DATA lines on the SD bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer This field is cleared in either of the following cases: • When the end field of the last data block is sent from the SD bus to uSDHC. • When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC module waits at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait to use the suspend / resume function. This field remains 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the write command • When writing to 1 to the Continue Request field in the Protocol Control register to continue a write transfer This field is cleared in either of the following cases:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• When the SD card releases Write Busy of the last data block, uSDHC also detects if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, uSDHC assumes the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and because of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This field is cleared when the DATA0 line is released. 0b - DATA line inactive 1b - DATA line active
1 CDIHB	Command Inhibit Data (DATA) This status field is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this field is 0, it indicates that uSDHC can issue the next SD / eMMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. **NOTE** The SD host driver can save registers for a suspend transaction after this field has changed from 1 to 0. 0b - Can issue command that uses the DATA line 1b - Cannot issue command that uses the DATA line
0 CIHB	Command inhibit (CMD) If this status bit is 0, it indicates that the CMD line is not in use and uSDHC can issue a SD / eMMC command using the CMD line. This field is set also immediately after the Transfer Type register is written. This field is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, commands using only the CMD line can be issued if this field is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If uSDHC cannot issue the command because of a command conflict error (see Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this field remains 1 and the Command Complete is not set. The Status of issuing an auto CMD12 does not show on this field. 0b - Can issue command using only CMD line 1b - Cannot issue command

65.6.1.12 Protocol Control (PROT_CTRL)

Offset

Register	Offset
PROT_CTRL	28h

Function

This register controls three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether uSDHC issues a Suspend command or the SD card accepts the Suspend command.

- If the host driver does not issue a Suspend command, the Continue request is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card accepts it, a Resume command is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card does not accept it, the Continue request is used to restart the transfer.

Any time stop at block gap request stops the data transfer, the host driver waits for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the host driver clears the Stop At Block Gap Request before or simultaneously.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	NON_EXA...	BURST_LEN_EN			WECRM	WECINS	WECINT	Reserved			RD_DONE...	IABG	RWCTL	CREQ	SABGREQ
W																
Reset	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						DMASEL		Reserved	Reserved	EMODE		D3CD	DTW		Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Fields

Field	Function
31 —	Reserved Always write as 0
30 NON_EXACT_BLK_RD	Non-exact block read Current block read is non-exact block read. It is only used for SDIO. 0b - The block read is exact block read. Host driver does not need to issue abort command to terminate this multi-block read. 1b - The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read.
29-27 BURST_LEN_EN	BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP This field is used to enable or disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLEs on the AXI side. 1xxb - Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	x1xb - Burst length is enabled for INCR4 / INCR8 / INCR16. xx1b - Burst length is enabled for INCR.
26 WECRM	Wakeup event enable on SD card removal This field enables a wakeup event, via a card removal, in the Interrupt Status register. FN_WUS (Wakeup Support) in CIS does not affect this field. When this field is set, the Card Removal Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Removal Status and uSDHC interrupt. 0b - Disables wakeup event enable on SD card removal 1b - Enables wakeup event enable on SD card removal
25 WECINS	Wakeup event enable on SD card insertion This field enables a wakeup event, via a card insertion, in the Interrupt Status register. FN_WUS (Wakeup Support) in CIS does not affect this field. When this field is set, the Card Insertion Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Insertion Status and uSDHC interrupt. 0b - Disable wakeup event enable on SD card insertion 1b - Enable wakeup event enable on SD card insertion
24 WECINT	Wakeup event enable on card interrupt This field enables a wakeup event, via a card interrupt, in the Interrupt Status register. This field can be set to 1 if FN_WUS (Wakeup Support) in CIS is set to 1. When this field is set, the Card Interrupt Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Interrupt Status and uSDHC interrupt. 0b - Disables wakeup event enable on card interrupt 1b - Enables wakeup event enable on card interrupt
23-21 —	Reserved Always write as 3'b100
20 RD_DONE_NO_8CLK	Read performed number 8 clock According to the SD/eMMC spec, for read data transaction, 8 clocks are needed after the end field of the last data block. So, by default(RD_DONE_NO_8CLK=0), eight clocks are active after the end field of the last read data transaction. However, these 8 clocks should not be active if user wants to use stop at block gap (include the auto stop at block gap in boot mode) feature for read and the RWCTL field (bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid these 8 clocks. Otherwise, the device might send extra data to uSDHC while uSDHC ignores these data. In a summary, this field should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.
19	Interrupt at block gap

Table continues on the next page...

Table continued from the previous page...

Field	Function
IABG	This field is valid only in 4-bit mode, of SDIO, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If SDIO cannot signal an interrupt during a multiple block transfer, this field should be set to 0 to avoid an inadvertent interrupt. When the host driver detects an SDIO insertion, it sets this field according to the CCCR of the card. 0b - Disables interrupt at block gap 1b - Enables interrupt at block gap
18 RWCTL	Read wait control The read wait function provided by this field is optional for SDIO. If the card supports read wait, set this field to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise, uSDHC has to stop the SD clock to hold read data, which restricts commands generation. When the host driver detects an SDIO insertion, it sets this field according to the CCCR of the card. If the card does not support read wait, this field should never be set to 1; otherwise, DATA line conflicts might occur. If this field is set to 0, stop at block gap during read operation is also supported, but uSDHC stops the SD clock to pause reading operation. 0b - Disables read wait control and stop SD clock at block gap when SABGREQ field is set 1b - Enables read wait control and assert read wait without stopping SD clock at block gap when SABGREQ field is set
17 CREQ	Continue request This field is used to restart a transaction which was stopped using the stop at block gap request. When a suspend operation is not accepted by the card, it is also by setting this field to restart the paused transfer. To cancel stop at the block gap, set stop at block gap request to 0 and set this field to 1 to restart the transfer. The uSDHC module automatically clears this field, therefore it is not necessary for the host driver to set this field to 0. If both stop at block gap request and this field are 1, the continue request is ignored. 0b - No effect 1b - Restart
16 SABGREQ	Stop at block gap request This field is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the transfer complete is set to 1, indicating a transfer completion, the host driver leaves this field set to 1. Clearing both the stop at block gap request and continue request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC module supports the stop at block gap request for write transfers, but for read transfers it requires that SDIO support read wait. Therefore, the host driver does not set this field during read transfers unless SDIO supports Read Wait and has set the read wait control to 1; otherwise, uSDHC stops the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the host driver writes data to the Data Port register, the host driver sets this field after all block data is written. If this field is set to 1, the host driver does not write data to the Data Port register after a block is sent. Once this field is set, the host driver does not clear this field before the Transfer Complete field in Interrupt Status register is set, otherwise uSDHC's behavior is undefined. This field effects read transfer active, write transfer active, DATA Line Active and Command Inhibit (DATA) in the Present State register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Transfer 1b - Stop
15-10 —	Reserved
9-8 DMASEL	DMA select This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00b - No DMA or simple DMA is selected. 01b - ADMA1 is selected. 10b - ADMA2 is selected. 11b - Reserved
7 —	Reserved
6 —	Reserved
5-4 EMODE	Endian mode This field supports all three endian modes in data transfer. See Data buffer for more details. 00b - Big endian mode 01b - Half word big endian mode 10b - Little endian mode 11b - Reserved
3 D3CD	DATA3 as card detection pin If this field is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 might set the card into the SPI mode, which uSDHC does not support. 0b - DATA3 does not monitor card insertion 1b - DATA3 as card detection pin
2-1 DTW	Data transfer width This field selects the data width of the SD bus for a data transfer. The host driver sets it to match the data width of the card. Possible data transfer width is 1-bit, 4-bits or 8-bits. 00b - 1-bit mode 01b - 4-bit mode 10b - 8-bit mode

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Reserved
0 —	Reserved

65.6.1.13 System Control (SYS_CTRL)

Offset

Register	Offset
SYS_CTRL	2Ch

Function

This register provides control of the system. See detail in the field description.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			RSTT	INITA	RSTD	RSTC	RSTA	IPP_RST...	RST_FIFO	Reserved	0	DTOCV			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SDCLKFS								DVS				Reserved			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Fields

Field	Function
31-29 —	Reserved
28 RSTT	Reset tuning When set this field to 1, it resets tuning circuit. After tuning circuits are reset, bit value is 0. Clearing execute_tuning field in AUTOCMD12_ERR_STATUS also sets this field to 1 to reset tuning circuit
27 INITA	Initialization active

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When this field is set, 80 SD-clocks are sent to the card. After the 80 clocks are sent, this field is self cleared. This field is very useful during the card power-up period when 74 SD-clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this field is already 1 has no effect. Writing 0 to this field at any time has no effect. When either of the CIHB and CDIHB fields in the Present State register are set, writing 1 to this field is ignored (that is, when command line or data lines are active, write to this field is not allowed). On the other-hand, when this field is set, that is, during initialization active period, it is allowed to issue command, and the command bit stream appears on the CMD pad after all 80 clock cycles are done. So, when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs to send 80 cycles to the card and does not want to wait till this field is self cleared.
26 RSTD	Software reset for data line Only part of the data circuit is reset. DMA circuit is also reset. After this field is set, the software waits for self-clear. The following registers and bits are cleared by this field: • Data Port register — Buffer is cleared and initialized • Present State register — Buffer read enable — Buffer write enable — Read transfer active — Write transfer active — DATA line active — Command Inhibit (DATA) • Protocol Control register — Continue request • Interrupt Status register — Buffer read ready — Buffer write ready — DMA interrupt — Block gap event — Transfer complete **NOTE** When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set. 0b - No reset 1b - Reset
25	Software reset for CMD line

Table continues on the next page...

Table continued from the previous page...

Field	Function
RSTC	Only part of the command circuit is reset. After this field is set, the software waits for self-clear. The following registers and bits are cleared by this field: • Present State Register — Command Inhibit (CMD) • Interrupt Status Register — Command Complete 0b - No reset 1b - Reset
24 RSTA	Software reset for all This reset affects the entire host controller except for the card detection circuit. RSTA resets all the registers that can be reset by RSTC/RSTD. During its initialization, the host driver is set this field to 1 to reset uSDHC. The uSDHC module resets this field to 0 when the capabilities registers are valid and the host driver can read them. Additional use of Software Reset For All does not affect the value of the capabilities registers. After this field is set, it is recommended that the host driver reset the external card and re-initialize it. After this field is set, the software should wait for self-clear. In tuning process, after every CMD19 is finished, this field is set to retest uSDHC. **NOTE** When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set. 0b - No reset 1b - Reset
23 IPP_RST_N	Hardware reset This field's value is output to card through pad directly to hardware reset pin of the card if the card supports this feature.
22 RST_FIFO	Reset the async FIFO Reset the async FIFO between card interface and the internal logic. After this field is set, the software waits for self-clear.
21 —	Reserved
20 —	Reserved
19-16 DTOCV	Data timeout counter value This value determines the interval by which DAT line timeouts are detected. See the Data Timeout Error field in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency is generated by dividing the base clock SDCLK value by this value.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The host driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 0000b - SDCLK x 2^{32} 0001b - SDCLK x 2^{33} 0010b - SDCLK x 2^{18} 0011b - SDCLK x 2^{19} 1101b - SDCLK x 2^{29}, recommend to use for supported speed modes except SDR104 mode 1110b - SDCLK x 2^{30}, recommend to use for SDR104 mode
15-8 SDCLKFS	SDCLK frequency select This field is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this field holds the prescaler (of this register) and divisor (next register) of the Base Clock Frequency register. In Single Data Rate mode (DDR_EN field of MIXERCTRL is '0') Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4 01h) Base clock divided by 2 00h) Base clock divided by 1 While in Dual Data Rate mode (DDR_EN field of MIXERCTRL is '1') Only the following settings are allowed: 80h) Base clock divided by 512 40h) Base clock divided by 256 20h) Base clock divided by 128 10h) Base clock divided by 64 08h) Base clock divided by 32 04h) Base clock divided by 16 02h) Base clock divided by 8 01h) Base clock divided by 4 00h) Base clock divided by 2

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When the software changes the DDR_EN field, SDCLKFS might need to be changed also. In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h yields 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD clock after reset is 375 kHz. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. If setting SDCLKFS and DVS can generate the same clock frequency,(for example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.), SDCLKFS is highly recommended. For SDR104 mode, if DLL is to be used, make sure SDCLKFS is used so that REF_CLK, generated for PARTS DLL, is at the same frequency as SD_CLK.
7-4 DVS	Divisor This field is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. Before changing clock divisor value (SDCLKFS or DVS), Host driver should make sure the SDSTB field is high. The settings are as follows: 0000b - Divide-by-1 0001b - Divide-by-2 1110b - Divide-by-15 1111b - Divide-by-16
3-0 —	Reserved Always write as 1.

65.6.1.14 Interrupt Status (INT_STATUS)

Offset

Register	Offset
INT_STATUS	30h

Function

An interrupt is generated when the normal interrupt signal enable is enabled and at least one of the status fields is set to 1. For all fields, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For card interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the card driver services the interrupt condition; otherwise, the CINT field is asserted again.

The table below shows the relationship between the command timeout error and the command complete.

Table 543. uSDHC status for command timeout error/command complete bit combinations

Command complete	Command timeout error	Meaning of the status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the transfer complete and the data timeout error.

Table 544. uSDHC status for data timeout error/transfer complete bit combinations

Transfer complete	Data timeout error	Meaning of the status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data transfer complete

The table below shows the relationship between the command CRC error and command timeout error.

Table 545. uSDHC status for command CRC error/command timeout error bit combinations

Command CRC error	Command timeout error	Meaning of the status
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0		DMAE	0	TNE	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				W1C		W1C		W1C		W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERR_INT...	TP	0	RTE	0	0		CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W		W1C		W1C				W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 DMAE	DMA error Occurs when an Internal DMA transfer has failed. This field is set to 1, when some error occurs in the data transfer. This error can be caused by either simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Because any error corrupts the whole data block, the host driver restarts the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 0b - No error 1b - Error
27 —	Reserved
26 TNE	Tuning error: (only for SD3.0 SDR104 mode) This field is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, host driver needs to abort a command executing and perform tuning. This field is relevant in combination with the bus error only and must be cleared prior to all the write and read transactions.
25 —	Reserved
24 AC12E	Auto CMD12 error

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Occurs when detecting that one of the fields in the Auto CMD12 Error Status register has changed from 0 to 1. This field is set to 1, not only when the errors in Auto CMD12 occur, but also, when the Auto CMD12 is not executed due to the previous command error. 0b - No error 1b - Error
23 —	Reserved
22 DEBE	Data end bit error Occurs either when detecting 0 at the end field position of read data that uses the DATA line, or at the end field position of the CRC. This field is not asserted in tuning process. 0b - No error 1b - Error
21 DCE	Data CRC error Occurs when detecting a CRC error when transferring read data that uses the DATA line, or when detecting the Write CRC status having a value other than 010. This field is not asserted in tuning process. 0b - No error 1b - Error
20 DTOE	Data timeout error Occurs when detecting one of following time-out conditions. • Busy time-out for R1b, R5b type • Busy time-out after Write CRC status • Read Data time-out. This field is not asserted in tuning process. 0b - No error 1b - Time out
19 CIE	Command index error Occurs if a command index error occurs in the command response. This field is not asserted in tuning process. 0b - No error 1b - Error
18	Command end bit error

Table continues on the next page...

Table continued from the previous page...

Field	Function
CEBE	Occurs when detecting that the end field of a command response is 0. This field is not asserted in tuning process. 0b - No error 1b - End bit error generated
17 CCE	Command CRC error Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this field is set when detecting a CRC error in the command response. • The uSDHC module detects a CMD line conflict by monitoring the CMD line when a command is issued. If uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then uSDHC aborts the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error should also be set to 1 to distinguish CMD line conflict. This field is not asserted in tuning process. 0b - No error 1b - CRC error generated
16 CTOE	Command timeout error Occurs only if no response is returned within 64 SDCLK cycles from the end field of the command. If uSDHC detects a CMD line conflict, in which case a Command CRC Error is also be set (as shown in Interrupt Status (INT_STATUS)), this field is set without waiting for 64 SDCLK cycles. This is because the command is aborted by uSDHC. This field is not asserted in tuning process. 0b - No error 1b - Time out
15 ERR_INT_STATUS	Error Interrupt Status This bit is set when any of the error status bit DMAE, AC12E, DEBE, DCE, DTOE, CIE, CEBE, CCE and CTOE is set.
14 TP	Tuning pass:(only for SD3.0 SDR104 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.
13 —	Reserved
12 RTE	Re-tuning event: (only for SD3.0 SDR104 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. host controller requests host driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Re-tuning is not required. 1b - Re-tuning should be performed.
11 —	Reserved
10-9 —	Reserved
8 CINT	Card interrupt This status field is set when an interrupt signal is detected from the external card. In 1-bit mode, uSDHC detects the Card Interrupt without the SD clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from SDIO and the interrupt to the host system. Writing this field to 1 can clear this field, but as the interrupt source from SDIO does not clear, this field is set again. to clear this field, it is required to reset the interrupt source from the external card followed by a writing 1 to this field. When this status has been set, and the host driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the host system. After completion of the card interrupt service (It should reset the interrupt sources in SDIO and the interrupt signal might not be asserted), write 1 to clear this field, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again. 0b - No card interrupt 1b - Generate card interrupt
7 CRM	Card removal This status field is set if the Card Inserted field in the Present State register changes from 1 to 0. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if no card is inserted. to leave it cleared, clear the Card Removal Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or inserted 1b - Card removed
6 CINS	Card insertion This status field is set if the Card Inserted field in the Present State register changes from 0 to 1. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if a card is inserted. to leave it cleared, clear the Card Inserted Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or removed 1b - Card inserted

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 BRR	Buffer read ready This status field is set if the Buffer Read Enable field, in the Present State register, changes from 0 to 1. See the Buffer Read Enable field in the Present State register for additional information. This field indicates that cmd19 is finished in tuning process. 0b - Not ready to read buffer 1b - Ready to read buffer
4 BWR	Buffer write ready This status field is set if the Buffer Write Enable field, in the Present State register, changes from 0 to 1. See the Buffer Write Enable field in the Present State register for additional information. 0b - Not ready to write buffer 1b - Ready to write buffer
3 DINT	DMA interrupt Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this field does not be set. Instead, the DMAE field is set. Either Simple DMA or ADMA finishes data transferring, this field is set. 0b - No DMA interrupt 1b - DMA interrupt is generated.
2 BGE	Block gap event If the Stop At Block Gap Request field in the Protocol Control register is set, this field is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this field is not set to 1. In the case of a Read Transaction: This field is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD bus timing). The Read Wait must be supported to use this function. In the case of Write Transaction: This field is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 0b - No block gap event 1b - Transaction stopped at block gap
1 TC	Transfer complete This field is set when a read or write transfer is completed. In the case of a Read Transaction: This field is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the host system). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request field in the Protocol Control register (after valid data has been read to the host system). In the case of a Write Transaction: This field is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as

Table continues on the next page...

Table continued from the previous page...

Field	Function
	specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request field in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this field is set when busy is deasserted. This field is not asserted in tuning process. 0b - Transfer does not complete 1b - Transfer complete
0 CC	Command complete This field is set when you receive the end field of the command response (except auto CMD12). See the Command Inhibit (CMD) in the Present State register. This field is not asserted in tuning process. 0b - Command not complete 1b - Command complete

65.6.1.15 Interrupt Status Enable (INT_STATUS_EN)

Offset

Register	Offset
INT_STATUS_EN	34h

Function

Setting any bit in this register to 1 enables the corresponding interrupt status be requested to the system. If any bit is set to 0, the corresponding interrupt request gets blocked.

- Depending on IABG field setting, uSDHC might be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There are some delays on the Card Interrupt, asserted from the card, to the time the host system is informed.
- To detect a CMD line conflict, the host driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	Reserved		DMAESEN	0	TNESEN	0	AC12ESEN	0	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TPSEN	0	RTESEN	0	Reserved		CINTSEN	CRMSEN	CINSSEN	BRRSEN	BWRSEN	DINTSEN	BGESEN	TCSEN	CCSEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 DMAESEN	DMA error status enable 0b - Masked 1b - Enabled
27 —	Reserved
26 TNESEN	Tuning error status enable 0b - Masked 1b - Enabled
25 —	Reserved
24 AC12ESEN	Auto CMD12 error status enable 0b - Masked 1b - Enabled
23 —	Reserved
22 DEBESEN	Data end bit error status enable 0b - Masked

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enabled
21 DCESEN	Data CRC error status enable 0b - Masked 1b - Enabled
20 DTOESEN	Data timeout error status enable 0b - Masked 1b - Enabled
19 CIESEN	Command index error status enable 0b - Masked 1b - Enabled
18 CEBESEN	Command end bit error status enable 0b - Masked 1b - Enabled
17 CCESEN	Command CRC error status enable 0b - Masked 1b - Enabled
16 CTOESEN	Command timeout error status enable 0b - Masked 1b - Enabled
15 —	Reserved
14 TPSEN	Tuning pass status enable 0b - Masked 1b - Enabled
13 —	Reserved
12 RTESEN	Re-tuning event status enable 0b - Masked 1b - Enabled
11 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
10-9 —	Reserved
8 CINTSEN	Card interrupt status enable If this field is set to 0, uSDHC clears the interrupt request to the system. The Card Interrupt detection is stopped when this field is cleared and restarted when this field is set to 1. The host driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this field again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 0b - Masked 1b - Enabled
7 CRMSEN	Card removal status enable 0b - Masked 1b - Enabled
6 CINSSEN	Card insertion status enable 0b - Masked 1b - Enabled
5 BRRSEN	Buffer read ready status enable 0b - Masked 1b - Enabled
4 BWRSEN	Buffer write ready status enable 0b - Masked 1b - Enabled
3 DINTSEN	DMA interrupt status enable 0b - Masked 1b - Enabled
2 BGESEN	Block gap event status enable 0b - Masked 1b - Enabled
1 TCSEN	Transfer complete status enable 0b - Masked 1b - Enabled
0 CCSEN	Command complete status enable 0b - Masked

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enabled

65.6.1.16 Interrupt Signal Enable (INT_SIGNAL_EN)

Offset

Register	Offset
INT_SIGNAL_EN	38h

Function

This register is used to select which interrupt status is indicated to the host system as the interrupt. These status fields all share the same interrupt lines. Setting any of these fields to 1 enables interrupt generation. The corresponding Status register field generates an interrupt when the corresponding interrupt signal enable field is set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	Reserved		DMAEIEN	0	TNEIEN	0	AC12EIEN	0	DEBEIEN	DCEIEN	DTOEIEN	CIEIEN	CEBEIEN	CCEIEN	CTOEIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TPIEN	0	RTEIEN	0	Reserved		CINTIEN	CRMIEN	CINSIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 DMAEIEN	DMA error interrupt enable 0b - Masked 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
27 —	Reserved
26 TNEIEN	Tuning error interrupt enable 0b - Masked 1b - Enabled
25 —	Reserved
24 AC12EIEN	Auto CMD12 error interrupt enable 0b - Masked 1b - Enabled
23 —	Reserved
22 DEBEIEN	Data end bit error interrupt enable 0b - Masked 1b - Enabled
21 DCEIEN	Data CRC error interrupt enable 0b - Masked 1b - Enabled
20 DTOEIEN	Data timeout error interrupt enable 0b - Masked 1b - Enabled
19 CIEIEN	Command index error interrupt enable 0b - Masked 1b - Enabled
18 CEBEIEN	Command end bit error interrupt enable 0b - Masked 1b - Enabled
17 CCEIEN	Command CRC error interrupt enable 0b - Masked 1b - Enabled
16	Command timeout error interrupt enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
CTOEIEN	0b - Masked 1b - Enabled
15 —	Reserved
14 TPIEN	Tuning Pass interrupt enable 0b - Masked 1b - Enabled
13 —	Reserved
12 RTEIEN	Re-tuning event interrupt enable 0b - Masked 1b - Enabled
11 —	Reserved
10-9 —	Reserved
8 CINTIEN	Card interrupt enable 0b - Masked 1b - Enabled
7 CRMIEN	Card removal interrupt enable 0b - Masked 1b - Enabled
6 CINSIEN	Card insertion interrupt enable 0b - Masked 1b - Enabled
5 BRRIEN	Buffer read ready interrupt enable 0b - Masked 1b - Enabled
4 BWRIEN	Buffer write ready interrupt enable 0b - Masked 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 DINTIEN	DMA interrupt enable 0b - Masked 1b - Enabled
2 BGEIEN	Block gap event interrupt enable 0b - Masked 1b - Enabled
1 TCIEN	Transfer complete interrupt enable 0b - Masked 1b - Enabled
0 CCIEN	Command complete interrupt enable 0b - Masked 1b - Enabled

65.6.1.17 Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)

Offset

Register	Offset
AUTOCMD12_ERR_STATUS	3Ch

Function

When the Auto CMD12 Error Status field in the Status register is set, the host driver checks this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in field 04-01. This register is valid only when the Auto CMD12 Error status field is set.

The table below shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 546. Relationship between command CRC error and command timeout error for auto CMD12

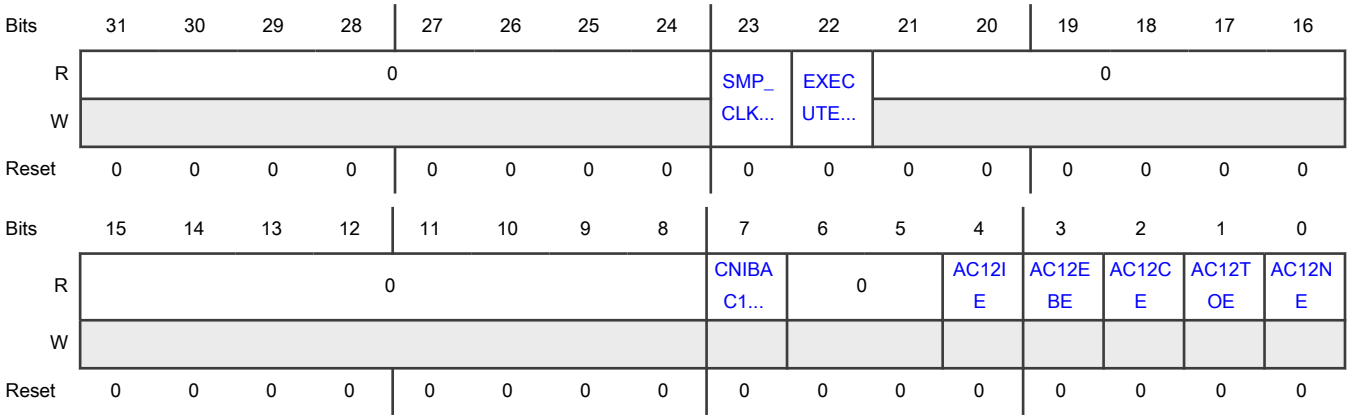
Auto CMD12 CRC error	Auto CMD12 timeout error	Type of error
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

- When uSDHC is going to issue an Auto CMD12
  - Set field 0 to 1 if the Auto CMD12 cannot be issued due to an error in the previous command
  - Set field 0 to 0 if the Auto CMD12 is issued
- At the end field of an Auto CMD12 response
  - Check errors correspond to fields 1-4
  - Set fields 1-4 corresponding to detected errors
  - Clear fields 1-4 corresponding to detected errors
- Before reading the Auto CMD12 Error Status field 7
  - Set field 7 to 1 if there is a command that can't be issued
  - Clear field 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, field 7 is sampled when the driver is not writing to the Command register. So, it is suggested to read this register only when the AC12E field in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error fields (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 SMP_CLK_SEL	Sample clock select This field is used to select sampling clock to receive CMD and DATA. This field is functional only when TUNING_CTRL[STD_TUNING_EN] is set. Otherwise, this field is reserved. This field is controlled by re-tuning procedure and is valid after the tuning procedure is complete (AUTOCMD12_ERR_STATUS[EXECUTE_TUNING]=0). A value of '1' means that tuning has been completed successfully and a value of '0' implies the tuning has failed. Writing '1' to this field is meaningless and ignored when the tuning has failed. The tuning circuit is reset by writing '0' to this field. This field can be cleared by setting AUTOCMD12_ERR_STATUS[EXECUTE_TUNING]=1. Once the tuning circuit is reset, it takes time to complete tuning sequence. Therefore, host driver should keep this field as 1 to perform

Table continues on the next page...

Table continued from the previous page...

Field	Function
	the re-tuning sequence in a short time. This field cannot be changed while the Host controller is receiving response or a read data block. 0b - Fixed clock is used to sample data 1b - Tuned clock is used to sample data
22 EXECUTE_TUNING	Execute tuning This field is used to start the tuning procedure. This field is functional only when TUNING_CTRL[STD_TUNING_EN] is set. Otherwise, this field is reserved. This field is set to start tuning procedure and automatically cleared when tuning procedure is completed. The result of tuning is indicated to AUTOCMD12_ERR_STATUS[SMP_CLK_SEL]. 0b - Tuning procedure is aborted 1b - Start tuning procedure
21-8 —	Reserved
7 CNIBAC12E	Command not issued by Auto CMD12 error Setting this field to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04-D01) in this register. 0b - No error 1b - Not issued
6-5 —	Reserved
4 AC12IE	Auto CMD12 / 23 index error Occurs if the command index error occurs in response to a command. 0b - No error 1b - Error, the CMD index in response is not CMD12/23
3 AC12EBE	Auto CMD12 / 23 end bit error Occurs when detecting that the end field of command response is 0 which should be 1. 0b - No error 1b - End bit error generated
2 AC12CE	Auto CMD12 / 23 CRC error Occurs when detecting a CRC error in the command response. 0b - No CRC error 1b - CRC error met in Auto CMD12/23 response
1	Auto CMD12 / 23 timeout error

Table continues on the next page...

Table continued from the previous page...

Field	Function
AC12TOE	Occurs if no response is returned within 64 SDCLK cycles from the end field of the command. If this field is set to1, the other error status fields (2-4) have no meaning. 0b - No error 1b - Time out
0 AC12NE	Auto CMD12 not executed If memory multiple block data transfer is not started, due to a command error, this field is not set because it is not necessary to issue an Auto CMD12. Setting this field to 1 means uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this field is set to 1, other error status fields (1-4) have no meaning. 0b - Executed 1b - Not executed

65.6.1.18 Host Controller Capabilities (HOST_CTRL_CAP)

Offset

Register	Offset
HOST_CTRL_CAP	40h

Function

This register provides the host driver with information specific to uSDHC implementation. The value in this register is the power-on-reset value and does not change with a software reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					VS18	VS30	VS33	SRS	DMAS	HSS	ADMAS	0	MBL		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		USE_TUN...	Reserved	Reserved				Reserved					DDR50_S...	SDR104_...	SDR50_S...
W																
Reset	1	0	1	1	0	1	0	0	0	0	0	0	0	1	1	1

Fields

Field	Function
31-27 —	Reserved
26 VS18	Voltage support 1.8 V This field depends on the host system ability. 0b - 1.8 V not supported 1b - 1.8 V supported
25 VS30	Voltage support 3.0 V This field depends on the host system ability. 0b - 3.0 V not supported 1b - 3.0 V supported
24 VS33	Voltage support 3.3 V This field depends on the host system ability. 0b - 3.3 V not supported 1b - 3.3 V supported
23 SRS	Suspend / resume support This field indicates whether uSDHC supports Suspend / Resume functionality. If this field is 0, the Suspend and Resume mechanism, as well as the read wait, are not supported, and the host driver does not issue either Suspend or Resume commands. 0b - Not supported 1b - Supported
22 DMAS	DMA support This field indicates whether uSDHC can use the internal DMA to transfer data between system memory and the data buffer directly. 0b - DMA not supported 1b - DMA supported
21 HSS	High speed support This field indicates whether uSDHC supports High Speed mode and the host system can supply a SD clock frequency from 25 MHz to 50 MHz. 0b - High speed not supported 1b - High speed supported
20 ADMAS	ADMA support This field indicates whether uSDHC supports the ADMA feature.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Advanced DMA not supported 1b - Advanced DMA supported
19 —	Reserved
18-16 MBL	Max block length This field indicates the maximum block size that the host driver can read and write to the buffer in uSDHC. The buffer transfers block size without wait cycles. 000b - 512 bytes 001b - 1024 bytes 010b - 2048 bytes 011b - 4096 bytes
15-14 —	Reserved
13 USE_TUNING_SDR50	Use Tuning for SDR50 This field is used to enable tuning for SDR50 mode 0b - SDR50 does not support tuning 1b - SDR50 supports tuning
12 —	Reserved
11-8 —	Reserved
7-3 —	Reserved
2 DDR50_SUPPORT	DDR50 support This field indicates support of DDR50 mode.
1 SDR104_SUPPORT	SDR104 support This field indicates support of SDR104 mode.
0	SDR50 support This field indicates support of SDR50 mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
SDR50_SUPPORT	

65.6.1.19 Watermark Level (WTMK_LVL)

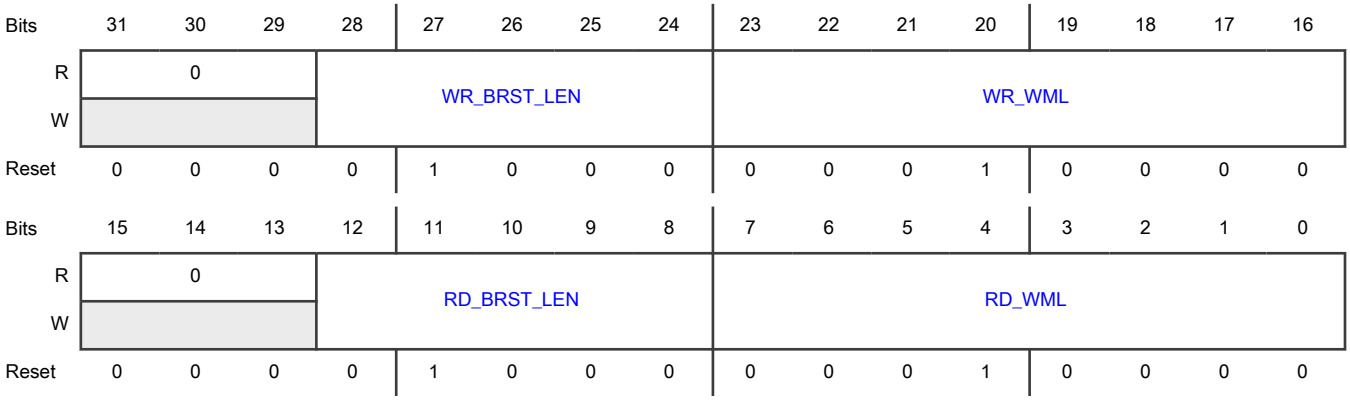
Offset

Register	Offset
WTMK_LVL	44h

Function

This register indicates configurability of write and read watermark levels (FIFO threshold). Their value can range from 1 to 128 words. Both write and read burst lengths are also configurable. Their value can range from 1 to 31 words.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-24 WR_BRST_LEN	Write burst length due to system restriction, the actual burst length might not exceed 16 This field indicates the number of words uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
23-16 WR_WML	Write watermark level

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field indicates the number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also, the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15-13 —	Reserved
12-8 RD_BRST_LEN	Read burst length due to system restriction, the actual burst length might not exceed 16 This field indicates the number of words uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
7-0 RD_WML	Read watermark level This field indicates the number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also, the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

65.6.1.20 Mixer Control (MIX_CTRL)

Offset

Register	Offset
MIX_CTRL	48h

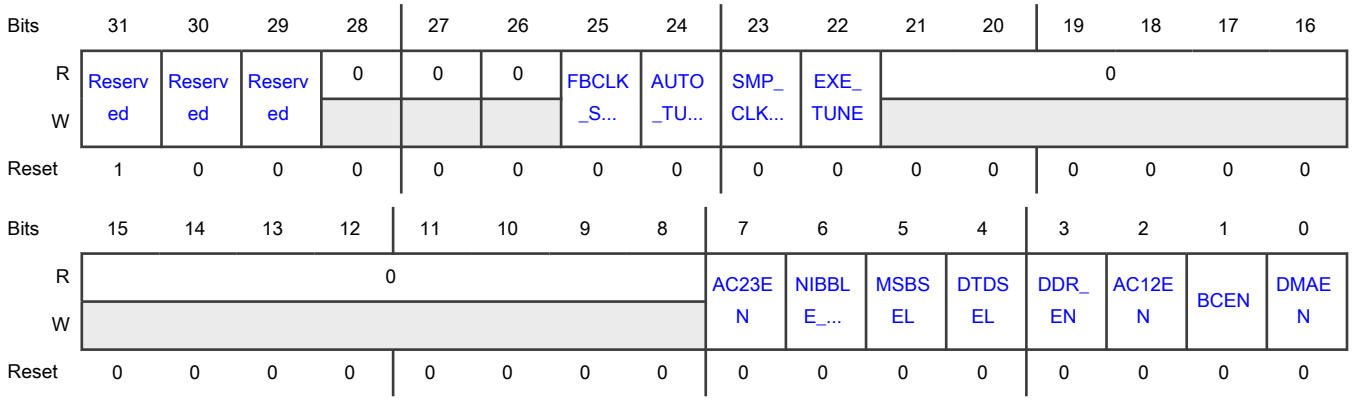
Function

This register is used to DMA and data transfer. To prevent data loss, the software should check if data transfer is active before writing this register. These fields are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

Table 547. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

Diagram



Fields

Field	Function
31 —	Reserved Always write as 1.
30 —	Reserved Always write as 0.
29 —	Reserved Always write as 0.
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 FBCLK_SEL	Feedback clock source selection (Only used for SD3.0, SDR104 mode) 0b - Feedback clock comes from the loopback CLK 1b - Feedback clock comes from the ipp_card_clk_out
24 AUTO_TUNE_EN	Auto tuning enable (Only used for SD3.0, SDR104 mode) 0b - Disable auto tuning 1b - Enable auto tuning
23 SMP_CLK_SEL	Clock selection When TUNING_CTRL[STD_TUNING_EN] is 0, this field is used to select Tuned clock or Fixed clock to sample data / cmd (Only used for SD3.0, SDR104 mode)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Fixed clock is used to sample data / cmd 1b - Tuned clock is used to sample data / cmd
22 EXE_TUNE	Execute tuning: (Only used for SD3.0, SDR104 mode) When TUNING_CTRL[STD_TUNING_EN] is 0, this field is set to 1 to indicate the host driver is starting tuning procedure. Tuning procedure is aborted by writing 0. 0b - Not tuned or tuning completed 1b - Execute tuning
21-8 —	Reserved
7 AC23EN	Auto CMD23 enable This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only. When this field is set to 1, the host controller issues a CMD23 automatically before issuing a command specified in the Command Register.
6 NIBBLE_POS	Nibble position indication This field indicates the nibble position in the DDR 4-bit mode. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.
5 MSBSEL	Multi / Single block select This field enables multiple block DATA line data transfers. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only. For any other commands, this field can be set to 0. If this field is 0, it is not necessary to set the Block Count register. See Command Transfer Type (CMD_XFR_TYP). 0b - Single block 1b - Multiple blocks
4 DTDSEL	Data transfer direction select This field defines the direction of DATA line data transfers. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only. The field is set to 1 by the host driver to transfer data from the SD card to uSDHC and is set to 0 for all other commands. If a CMD with the response data size cannot be divided by 64 bytes, is given without a STOP CMD, and is followed by a software reset, then you must write a 0 to DTDSEL before inserting a software reset. This is necessary to avoid any issue during the SD card reinitialization. Read multiple blocks will not be affected as it is required to have a STOP CMD. 0b - Write (Host to card) 1b - Read (Card to host)
3	Dual data rate mode selection

Table continues on the next page...

Table continued from the previous page...

Field	Function
DDR_EN	This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only.
2 AC12EN	Auto CMD12 enable Multiple block transfers for memory require a CMD12 to stop the transaction. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only. When this field is set to 1, uSDHC issues a CMD12 automatically when the last block transfer has completed. The host driver is not set this field to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, uSDHC ignores this field no matter it is set or not. 0b - Disable 1b - Enable
1 BCEN	Block count enable This field is used to enable the Block Count register, which is only relevant for multiple block transfers. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only. When this field is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 0b - Disable 1b - Enable
0 DMAEN	DMA enable This field enables DMA functionality. This field is read/write when VEND_SPEC[CMD_BYTE_EN] is disabled; otherwise, this field is read-only. If this field is set to 1, a DMA operation begins when the host driver sets the DPSEL field of this register. Whether the simple DMA or the advanced DMA is active depends on the DMA Select field of the Protocol Control register. 0b - Disable 1b - Enable

65.6.1.21 Force Event (FORCE_EVENT)

Offset

Register	Offset
FORCE_EVENT	50h

Function

This register is not a physically implemented register. Rather, it is an address at which the Interrupt Status register can be written if the corresponding field of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register sets the corresponding field of Interrupt Status register. A read from this register always results in 0's. to change the corresponding status fields in the Interrupt Status register, make sure to set IPGEN field in System Control Register so that peripheral clock is always active.

Forcing a card interrupt generates a short pulse on the DATA1 line, and the driver might treat this interrupt as a normal interrupt. The interrupt service routine might skip polling the card interrupt factor as the interrupt is self cleared.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
W	FEVTCINT			FEVTDMAE		FEVTTNE		FEVTAC1...		FEVTDEBE	FEVTDCE	FEVTDTOE	FEVTCIE	FEVTCEBE	FEVTCCE	FEVTCTOE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0	0		0	0	0	0	0
W									FEVTCNI...			FEVTAC1...	FEVTAC1...	FEVTAC1...	FEVTAC1...	FEVTAC1...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 FEVTCINT	Force event card interrupt Writing 1 to this field generates a short low-level pulse on the internal DATA1 line, as if a self-clearing interrupt was received from the external card. If enabled, the CINT field is set and the interrupt service routine might treat this interrupt as a normal interrupt from the external card.
30-29 —	Reserved
28 FEVTDMAE	Force event DMA error Forces the DMAE field of Interrupt Status register to be set
27 —	Reserved
26 FEVTTNE	Force tuning error Forces the TNE field of Interrupt Status register to be set
25 —	Reserved
24 FEVTAC12E	Force event Auto Command 12 error Forces the AC12E field of Interrupt Status register to be set
23 —	Reserved
22 FEVTDEBE	Force event data end bit error Forces the DEBE field of Interrupt Status register to be set

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 FEVTDCE	Force event data CRC error Forces the DCE field of Interrupt Status register to be set
20 FEVTDTOE	Force event data time out error Force the DTOE field of Interrupt Status register to be set
19 FEVTCIE	Force event command index error Forces the CCE field of Interrupt Status register to be set
18 FEVTCEBE	Force event command end bit error Forces the CEBE field of Interrupt Status register to be set
17 FEVTCCE	Force event command CRC error Forces the CCE field of Interrupt Status register to be set
16 FEVTCTOE	Force event command time out error Forces the CTOE field of Interrupt Status register to be set
15-8 —	Reserved
7 FEVTCNIBAC12E	Force event command not executed by Auto Command 12 error Forces the CNIBAC12E field in the Auto Command12 Error Status register to be set
6-5 —	Reserved
4 FEVTAC12IE	Force event Auto Command 12 index error Forces the AC12IE field in the Auto Command12 Error Status register to be set
3 FEVTAC12EBE	Force event Auto Command 12 end bit error Forces the AC12EBE field in the Auto Command12 Error Status register to be set
2 FEVTAC12CE	Force event auto command 12 CRC error Forces the AC12CE field in the Auto Command12 Error Status register to be set
1 FEVTAC12TOE	Force event auto command 12 time out error Forces the AC12TOE field in the Auto Command12 Error Status register to be set
0 FEVTAC12NE	Force event auto command 12 not executed Forces the AC12NE field in the Auto Command12 Error Status register to be set

65.6.1.22 ADMA Error Status (ADMA_ERR_STATUS)

Offset

Register	Offset
ADMA_ERR_STATUS	54h

Function

When an ADMA Error Interrupt has occurred, the ADMA error sates field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the host driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and does not generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

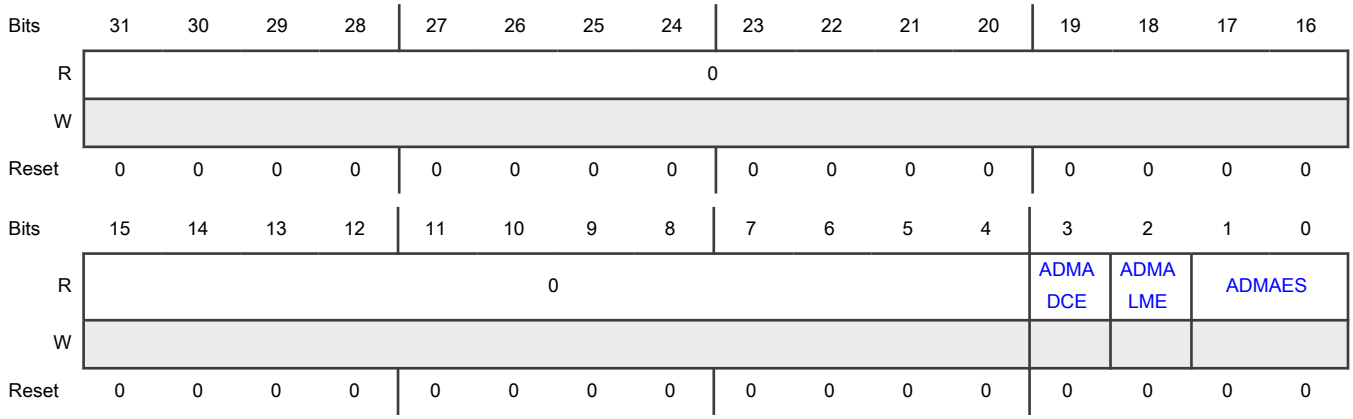
In case of a write operation, the host driver should use the ACMD22 to get the number of the written block, rather than using this information, because unwritten data might exist in the host controller.

The host controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The host driver can distinguish this error by reading the Valid field of the error descriptor.

Table 548. ADMA error state coding

D01-D00	ADMA error state (when error has occurred)	Contents of ADMA System Address register
00	ST_STOP (Stop DMA)	Holds the address of the next executable descriptor command
01	ST_FDS (Fetch descriptor)	Holds the valid descriptor address
10	ST_CADR (Change address)	No ADMA error is generated
11	ST_TFR (Transfer data)	Holds the address of the next executable descriptor command

Diagram



Fields

Field	Function
31-4 —	Reserved
3 ADMADCE	ADMA descriptor error This error occurs when invalid descriptor fetched by ADMA. 0b - No error 1b - Error
2 ADMALME	ADMA length mismatch error This error occurs in the following two cases: • While the block count enable is being set, the total data length specified by the descriptor table is different from that specified by the block count and block length. • Total data length cannot be divided by the block length. 0b - No error 1b - Error
1-0 ADMAES	ADMA error state (when ADMA error is occurred) This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. See ADMA Error Status (ADMA_ERR_STATUS) for more details.

65.6.1.23 ADMA System Address (ADMA_SYS_ADDR)

Offset

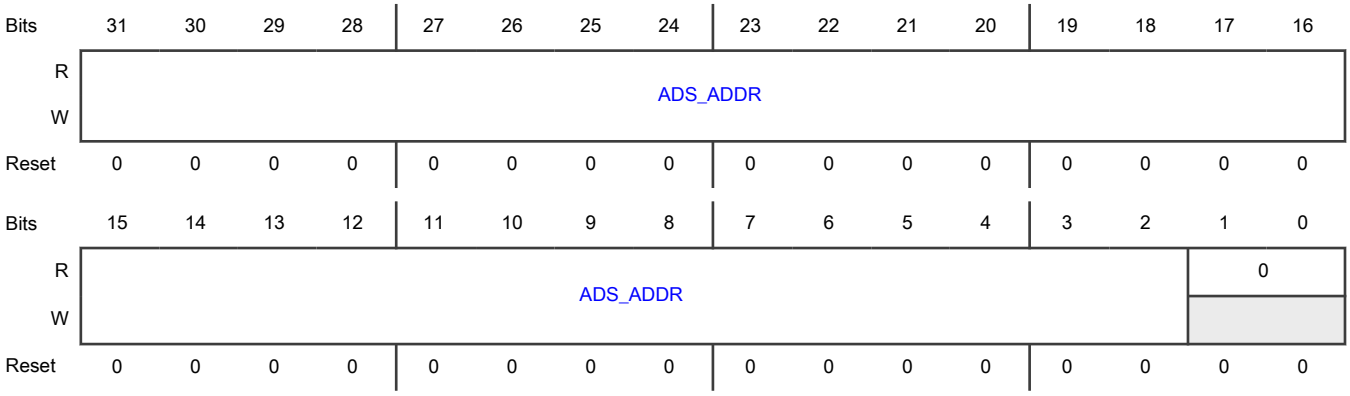
Register	Offset
ADMA_SYS_ADDR	58h

Function

This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the host driver sets the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register holds the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned.

Because this register supports dynamic address reflecting, when TC field is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC field is set. Such restriction is also listed in Software restrictions.

Diagram



Fields

Field	Function
31-2 ADS_ADDR	ADMA system address This field contains the physical system memory address used for ADMA transfers.
1-0 —	Reserved

65.6.1.24 DLL (Delay Line) Control (DLL_CTRL)

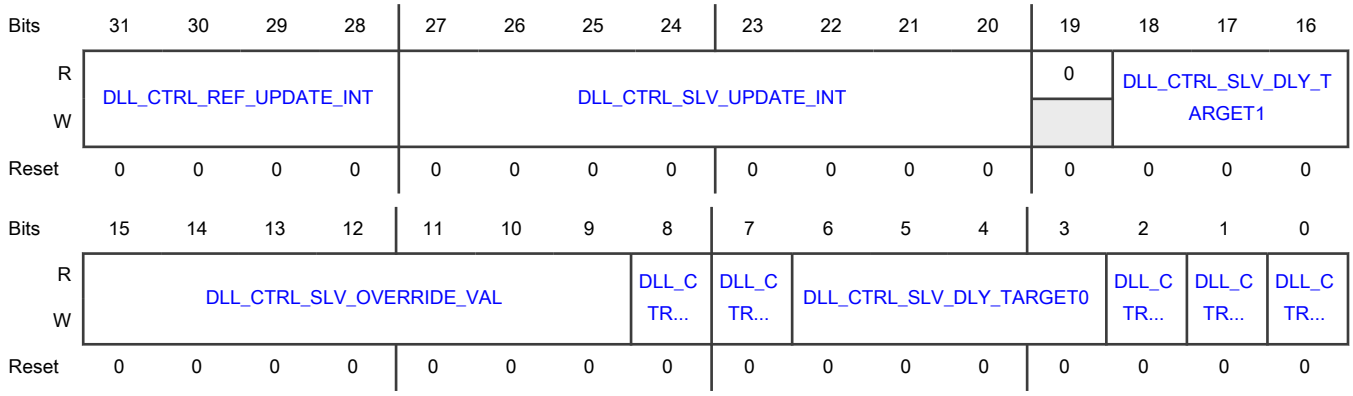
Offset

Register	Offset
DLL_CTRL	60h

Function

This register contains control fields for DLL.

Diagram



Fields

Field	Function
31-28 DLL_CTRL_REF_UPDATE_INT	DLL control loop update interval The interval cycle is (2 + REF_UPDATE_INT) * REF_CLOCK. By default, the DLL control loop updates every two REF_CLOCK cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that might effect the delay (such as voltage and temperature)
27-20 DLL_CTRL_SLV_UPDATE_INT	Slave delay line update interval If default 0 is used, it means 256 cycles of REF_CLOCK. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line also updates automatically when the reference DLL transitions to a locked state (from an un-locked state).
19 —	Reserved
18-16 DLL_CTRL_SLV_DLY_TARGET1	DLL slave delay target1 See DLL_CTRL_SLV_DLY_TARGET0 below.
15-9 DLL_CTRL_SLV_OVERRIDE_VAL	DLL slave override val When SLV_OVERRIDE = 1, this field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.
8 DLL_CTRL_SLV_OVERRIDE	DLL slave override Set this field to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE field. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0
7	DLL gate update

Table continues on the next page...

Table continued from the previous page...

Field	Function
DLL_CTRL_GATE_UPDATE	Set this field to 1 to prevent the DLL from updating (because when clock_in exists, glitches might appear during DLL updates). This field might be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6-3 DLL_CTRL_SLV_DLY_TARGET0	DLL slave delay target0 The delay target for uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is (({DLL_CTRL_SLV_DLY_TARGET1,DLL_CTRL_SLV_DLY_TARGET0} +1)* REF_CLOCK / 2) / 16 So the input read-clock can be delayed relative input data from (REF_CLOCK / 2) / 16 to REF_CLOCK * 4. **NOTE** For the restrictions of delay cell implementation, the delay target must be set between REF_CLOCK/16 and REF_CLOCK*2 when REF_CLOCK is running at 200 MHz. When REF_CLOCK frequency is slower than 100 MHz, the maximum delay target might not reach REF_CLOCK*2.
2 DLL_CTRL_SLV_FORCE_UPD	DLL slave delay line Setting this field to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line updates automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function might not work when uSDHC is working on data / cmd / response.
1 DLL_CTRL_RESET	DLL reset Setting this field to 1 force a reset on DLL. This causes the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, so to create a subsequent reset, RESET must be taken low and then asserted again.
0 DLL_CTRL_ENABLE	DLL and delay chain Set this field to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

65.6.1.25 DLL Status (DLL_STATUS)

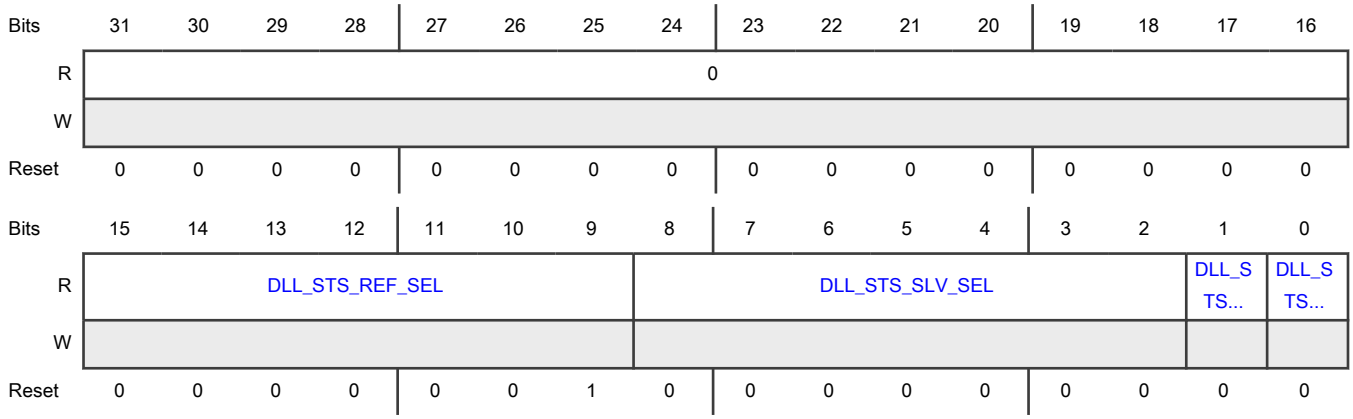
Offset

Register	Offset
DLL_STATUS	64h

Function

This register contains the DLL status information. All fields are read only and reads the same as the power-reset value.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-9 DLL_STS_REF_SEL	Reference delay line select taps This is encoded by 7 fields for 127 taps.
8-2 DLL_STS_SLV_SEL	Slave delay line select status This is the instant value generated from reference chain. Because the reference chain can only be updated when REF_CLOCK is detected, this value should be the right value to be updated when the reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

65.6.1.26 CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)

Offset

Register	Offset
CLK_TUNE_CTRL_STATUS	68h

Function

This register contains the Clock Tuning Control status information. This register is added to support SD3.0 UHS-I SDR104 mode.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRE_ERR	TAP_SEL_PRE							TAP_SEL_OUT				TAP_SEL_POST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NXT_ERR	DLY_CELL_SET_PRE							DLY_CELL_SET_OUT				DLY_CELL_SET_POST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PRE_ERR	PRE error PRE error which means the number of delay cells added on the feedback clock is too small. It is valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
30-24 TAP_SEL_PRE	TAP_SEL_PRE Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is enabled, TAP_SEL_PRE is updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.
23-20 TAP_SEL_OUT	Delay cells added on the feedback clock between CLK_PRE and CLK_OUT Reflects the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.
19-16 TAP_SEL_POST	Delay cells added on the feedback clock between CLK_OUT and CLK_POST Reflects the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.
15 NXT_ERR	NXT error NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
14-8 DLY_CELL_SET_PRE	delay cells on the feedback clock between the feedback clock and CLK_PRE Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-4 DLY_CELL_SET_OUT	Delay cells on the feedback clock between CLK_PRE and CLK_OUT Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.
3-0 DLY_CELL_SET_POST	Delay cells on the feedback clock between CLK_OUT and CLK_POST Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.

65.6.1.27 Vendor Specific Register (VEND_SPEC)

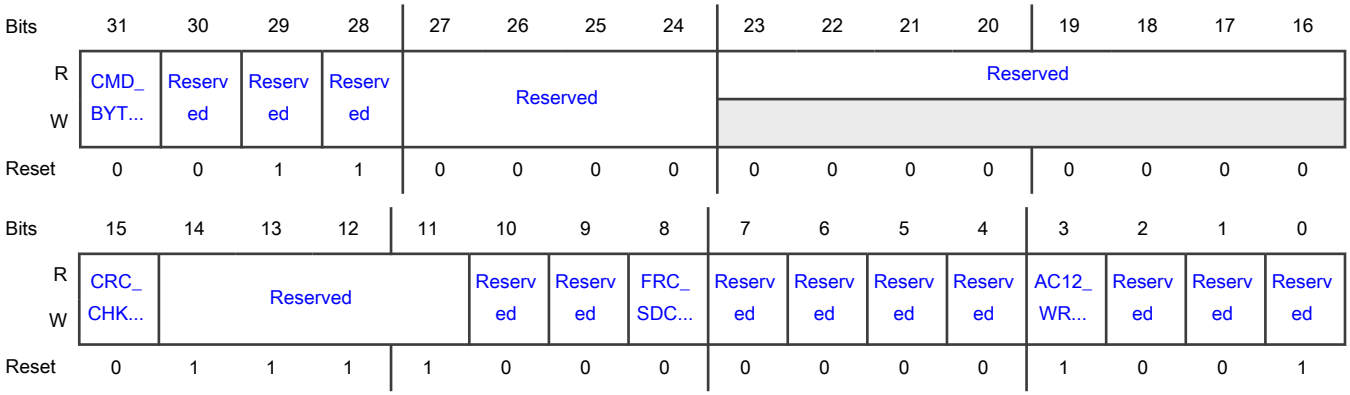
Offset

Register	Offset
VEND_SPEC	C0h

Function

This register contains the vendor specific control/status register.

Diagram



Fields

Field	Function
31 CMD_BYTE_EN	Register byte access for CMD_XFR_TYP This field controls the register byte access for Command Transfer Type (CMD_XFR_TYP). If this field is enabled, the register can be configured through byte write enable. IPS_bus can write only one byte once for one write operation. 0b - Disable. MIX_CTRL[7:0] is read/write and CMD_XFR_TYP[7:0] is read-only.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable. MIX_CTRL[7:0] is read-only and CMD_XFR_TYP[7:0] is read/write.
30 —	Reserved Always write as 0.
29 —	Reserved Always write as 1
28 —	Reserved Always write as 0.
27-24 —	Reserved Always write as 4'b0000.
23-16 —	Reserved Always write as 8'h00.
15 CRC_CHK_DIS	CRC Check Disable 0b - Check CRC16 for every read data packet and check CRC fields for every write data packet 1b - Ignore CRC16 check for every read data packet and ignore CRC fields check for every write data packet
14-11 —	Reserved Reserved
10 —	Reserved Always write as 0.
9 —	Reserved Always write as 0.
8 FRC_SDCLK_ON	Force CLK Force CLK output active Do not set this bit to 1 unless it is necessary. Also, make sure that this bit is cleared when uSDHC's clock is about to be changed (frequency change, clock source change, or delay chain tuning). 0b - CLK active or inactive is fully controlled by the hardware. 1b - Force CLK active
7 —	Reserved
6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 —	Reserved
4 —	Reserved
3 AC12_WR_CHKBUSY_EN	Check busy enable Check busy enable after auto CMD12 for write data packet 0b - Do not check busy after auto CMD12 for write data packet 1b - Check busy after auto CMD12 for write data packet
2 —	Reserved
1 —	Reserved
0 —	Reserved Always write as 0.

65.6.1.28 eMMC Boot (MMC_BOOT)

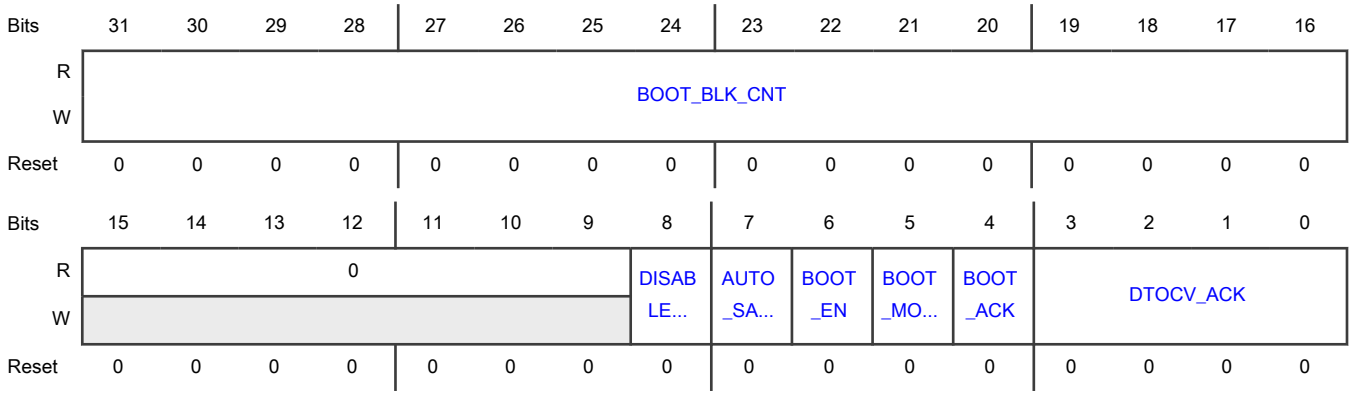
Offset

Register	Offset
MMC_BOOT	C4h

Function

This register contains the eMMC Fast Boot control register.

Diagram



Fields

Field	Function
31-16 BOOT_BLK_CNT	Stop At Block Gap value of automatic mode The value defines the Stop At Block Gap value of automatic mode. When received, card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Attributes Register, field 31 - 16 of 0x04.
15-9 —	Reserved
8 DISABLE_TIME_OUT	Time out **NOTE** When this field is set, there is no timeout check no matter whether BOOT_EN is set or not. 0b - Enable time out 1b - Disable time out
7 AUTO_SABG_EN	Auto stop at block gap During boot, enable auto stop at block gap function. This function is triggered, and host stops at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).
6 BOOT_EN	Boot enable Boot mode enable 0b - Fast boot disable 1b - Fast boot enable
5 BOOT_MODE	Boot mode Boot mode select 0b - Normal boot 1b - Alternative boot

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 BOOT_ACK	BOOT ACK Boot ACK mode select 0b - No ack 1b - Ack
3-0 DTOCV_ACK	Boot ACK time out Boot ACK time out counter value. 0000b - SDCLK x 2^14 0001b - SDCLK x 2^15 0010b - SDCLK x 2^16 0011b - SDCLK x 2^17 0100b - SDCLK x 2^18 0101b - SDCLK x 2^19 0110b - SDCLK x 2^20 0111b - SDCLK x 2^21 1110b - SDCLK x 2^28 1111b - SDCLK x 2^29

65.6.1.29 Vendor Specific 2 Register (VEND_SPEC2)

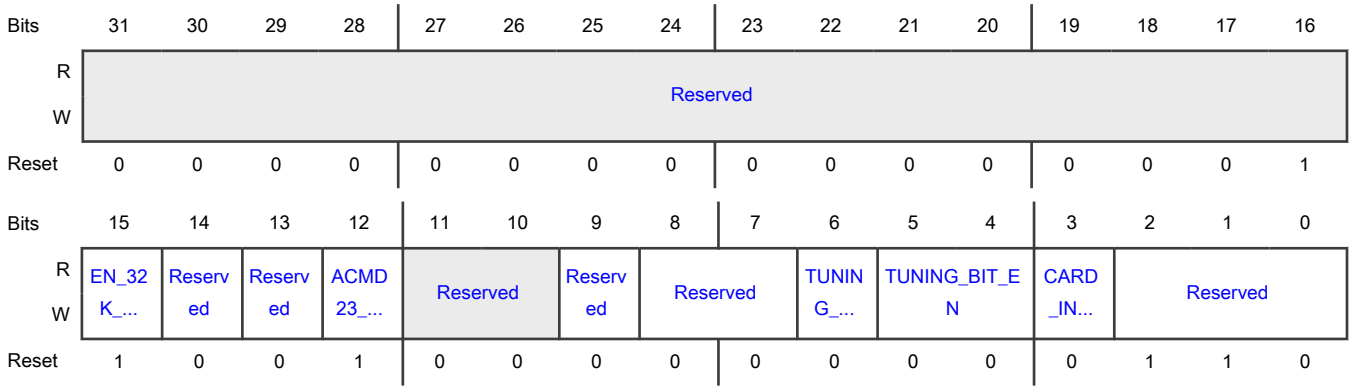
Offset

Register	Offset
VEND_SPEC2	C8h

Function

This register contains the vendor specific control 2 register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 EN_32K_CLK	Select the clock source for host card detection. It can use low power clock for card detection, by setting this bit to 1. 0b - Use the peripheral clock (ipg_clk) for card detection. 1b - Use the low power clock (ipg_clk_lp) for card detection.
14 —	Reserved
13 —	Reserved
12 ACMD23_ARGU2_EN	Argument2 register enable for ACMD23 0b - Disable 1b - Argument2 register enable for ACMD23 sharing with SDMA system address register. Default is enabled.
11-10 —	Reserved
9 —	Reserved
8-7 —	Reserved
6	Tuning command enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TUNING_CMD_EN	Enable the auto tuning circuit to check the CMD line. 0b - Auto tuning circuit does not check the CMD line. 1b - Auto tuning circuit checks the CMD line.
5-4 TUNING_BIT_EN	Tuning bit enable Enable the auto tuning circuit to check the DATA[7:0]. The value option is as the list below. **NOTE** In SDIO mode, auto-tuning only supports 1-bit (i.e. DATA[0]) tuning, and the 4-bit (i.e. DATA[3:0]) auto tuning is **not** supported for SDIO mode. SD mode supports 1-bit/4-bit (DATA[0] and DATA[3:0]) auto tuning. eMMC mode supports 1-bit/4-bit/8-bit (DATA[0], DATA[3:0] and DATA[7:0]) auto tuning. 00b - Enable Tuning circuit for DATA[3:0] 01b - Enable Tuning circuit for DATA[7:0] 10b - Enable Tuning circuit for DATA[0] 11b - Invalid
3 CARD_INT_D3_TEST	Card interrupt detection test This field is used only for debugging. 0b - Check the card interrupt only when DATA3 is high. 1b - Check the card interrupt by ignoring the status of DATA3.
2-0 —	Reserved

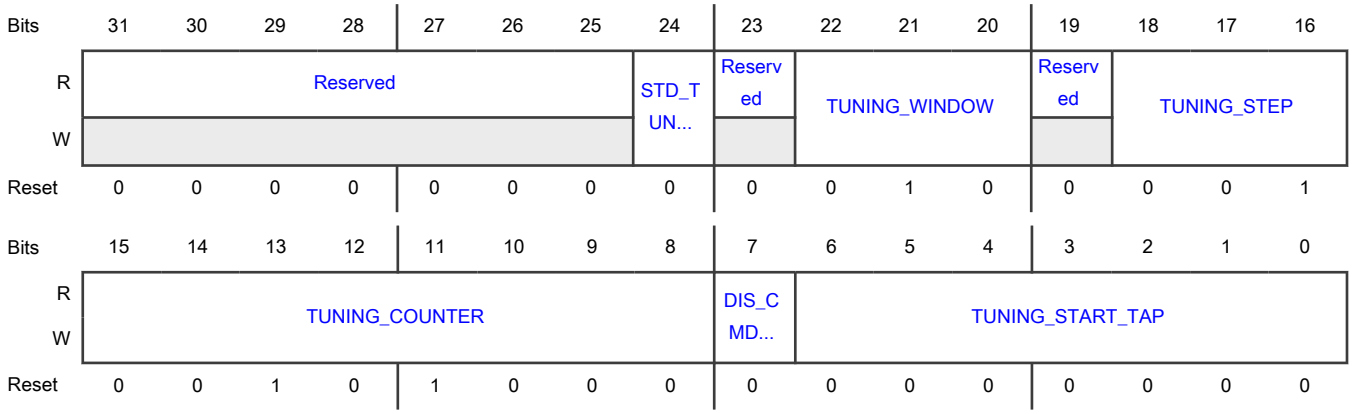
65.6.1.30 Tuning Control (TUNING_CTRL)

Offset

Register	Offset
TUNING_CTRL	CCh

Function
The register contains configuration of tuning circuit.

Diagram



Fields

Field	Function
31-25 —	Reserved
24 STD_TUNING_EN	Standard tuning circuit and procedure enable This field is used to enable standard tuning circuit and procedure.
23 —	Reserved
22-20 TUNING_WINDOW	Data window Select data window value for auto tuning
19 —	Reserved
18-16 TUNING_STEP	TUNING_STEP The increasing delay cell steps in tuning procedure.
15-8 TUNING_COUNTER	Tuning counter The MAX repeat CMD19 times in tuning procedure.
7 DIS_CMD_CHK_FOR_STD_TUNING	Disable command check for standard tuning Writing 1 to this field disables command check (command CRC, CMD end bit error, and CMD index error or CMD timeout error) for standard tuning flow after each tuning command is sent.
6-0	Tuning start

Table continues on the next page...

Table continued from the previous page...

Field	Function
TUNING_START_TAP	The start delay cell point when send first CMD19 in tuning procedure.

Chapter 66
Low-Power Flexible Communications Interface (LP_FLEXCOMM)

66.1 Chip-specific LP_FLEXCOMM information

Table 549. Reference links to related information

Topic	Related module	Reference
Full description	LP_FLEXCOMM	LP_FLEXCOMM
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See FLEXCOMMn_TRIG registers in INPUTMUX

66.1.1 Module instances

This chip has ten instances of LP_FLEXCOMM module, LP_FLEXCOMM0,1,2,3,4,5,6,7,8,9.

Each instance supports LPUART, LPSPI, and LPI2C functions. Each LPI2C, LPSPI, and LPUART instance is integrated within an LP_FLEXCOMM instance. Configure the LP_FLEXCOMM instance for I2C, SPI, and UART operation before using the corresponding LPI2C, LPSPI, and LPUART instance.

66.1.2 LP_FLEXCOMM configuration

When configured for LPSPI, LP_FLEXCOMM supports 8x32 bit FIFOs. When configured for LPI2C/LPUART, two 8x16 FIFOs are supported, one for each module. In any one of the ten instances, UART and I2C can be used simultaneously with limited functionality. All LP_FLEXCOMM instances support modbus and DMA End of Packet (EOP). There are two separate interrupts. The DMA requests and the trigger inputs/outputs are shared.

66.2 LP_FLEXCOMM

66.2.1 Overview

Each LP_FLEXCOMM interface provides one peripheral communications function from a choice of several, chosen by software. This chapter describes the overall LP_FLEXCOMM interface and how to choose the communications function. See the corresponding communications module chapter for details on each function provided.

66.2.1.1 Block diagram

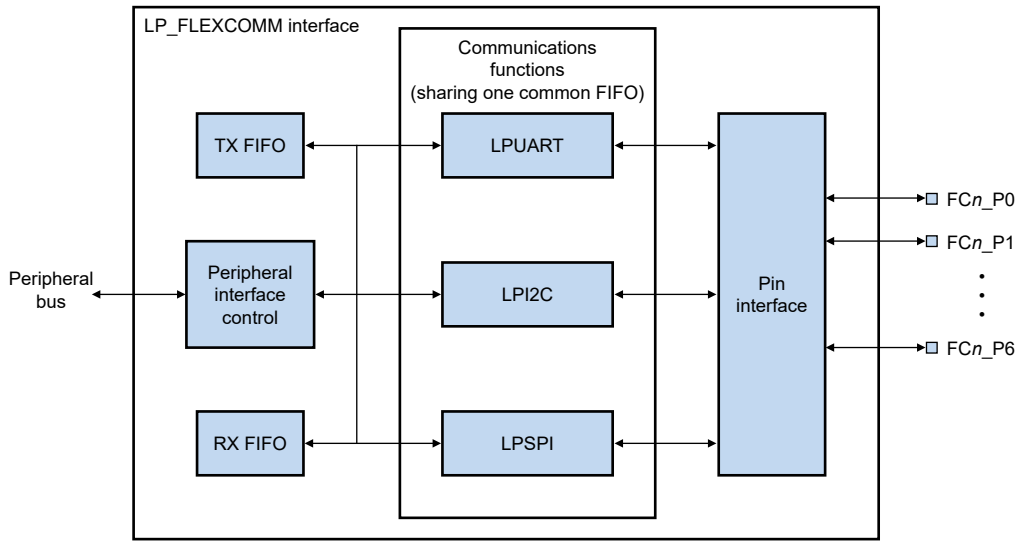


Figure 374. LP_FLEXCOMM interface block diagram

66.2.1.2 Features

Each LP_FLEXCOMM interface provides a choice of peripheral communications functions. You must choose one of them before you can configure and use the function.

- LPUART with asynchronous transmit and receive
- LPSPI controller or subordinate, with up to four peripheral chip selects
- LPI2C with separate controller and subordinate functions

LPUART and LPI2C can be selected at the same time with reduced interface.

A FIFO of 8 words is shared between the different interfaces. The LPI2C subordinate does not support a FIFO.

66.2.2 Functional description

The following sections describe the functional details of LP_FLEXCOMM.

66.2.2.1 Modes

This section describes the modes present in LP_FLEXCOMM.

66.2.2.1.1 Operating modes

Some communication options might not be available on a particular LP_FLEXCOMM interface. Reading PSELID shows which communication functions are available. You can select a specific supported function by writing the appropriate value to PSELID[PERSEL].

66.2.2.1.2 Low-power modes

The LP_FLEXCOMM interface follows the low-power mode operation of the selected communications module.

66.2.2.1.3 Debug mode

When the LP_FLEXCOMM interface is configured as LPSPI or LPI2C, you can configure it not to initiate a new transaction when the CPU is halted.

66.2.2.2 Clocking

The LP_FLEXCOMM interface requires a bus interface clock for the register interface of the selected communications module. It also requires a functional clock which generates the clocking for the peripheral communications interface. The clocks for unselected functions are internally gated.

66.2.2.3 Reset

The LP_FLEXCOMM interface receives a single reset that returns the entire module to its reset state. Out of reset, no peripheral communications function is selected, and the lock bit is clear.

66.2.2.4 Interrupts

The LP_FLEXCOMM interface generates a single interrupt from the configured peripheral communications function(s). When the LP_FLEXCOMM interface is configured for both LPUART and LPI2C functions, you can read the source of the interrupt from ISTAT.

66.2.2.5 DMA

The LP_FLEXCOMM interface generates two DMA requests:

- Transmit DMA Request
- Receive DMA Request (with End Of Packet support)

When the LP_FLEXCOMM interface is configured for both LPUART and LPI2C functions, you must enable DMA requests for only one of them.

66.2.2.6 Triggers

The LP_FLEXCOMM interface receives a single input trigger that is routed to all internal modules. When configured for both LPUART and LPI2C functions, you must enable the input trigger for only one of them.

The LP_FLEXCOMM interface generates 4 output triggers:

Output	LPSPI	LPUART	LPI2C	LPUART+LPI2C
Trigger 0	Frame	TX Word	Controller EOP	Controller EOP
Trigger 1	Word	RX Word	Subordinate EOP	Subordinate EOP
Trigger 2	—	RX Idle	Controller Busy	RX Idle
Trigger 3	—	TX Data	Bus Busy	TX Data

66.2.3 External signals

This table describes the LP_FLEXCOMM interface module signals. Pin usage for a specific communications function is described in the corresponding chapter of each communications module.

Signal	LPSPI	LPUART	LPI2C	LPUART+LPI2C
FC_P0	SDO/DATA[0]	RXD	SDA	SDA
FC_P1	SCK	TXD	SCL	SCL
FC_P2	SDI/DATA[1]	RTS_b	SCLS	TXD
FC_P3	PCS[0]	CTS_b	SDAS	RXD
FC_P4	PCS[3]/DATA[3]	DSR_b	HREQ_b	CTS_b
FC_P5	PCS[2]/DATA[2]	DTR_b	—	RTS_b
FC_P6	PCS[1]/HREQ	DCD_b	—	HREQ_b

66.2.4 Initialization

To initialize a LP_FLEXCOMM interface:

1. Enable the bus interface clock for the LP_FLEXCOMM interface.
2. Select a clock source for the functional clock of the selected communications module.
3. Select the function by writing the appropriate value to PSELID[PERSEL].
4. See the specific communications module chapter to complete the configuration.

66.2.5 Application information

LP_FLEXCOMM interface applications mirror those of the selected communications module.

66.2.6 Memory Map and register definition

This section includes the LP_FLEXCOMM interface module memory map and detailed descriptions of all registers.

The LP_FLEXCOMM interface supports 8-bit, aligned 16-bit, and aligned 32-bit register accesses, unless otherwise noted in the register description of the selected peripheral function.

66.2.6.1 LP_FLEXCOMM register descriptions

66.2.6.1.1 LP_FLEXCOMM memory map

LP_FLEXCOMM0 base address: 4009_2000h

LP_FLEXCOMM1 base address: 4009_3000h

LP_FLEXCOMM2 base address: 4009_4000h

LP_FLEXCOMM3 base address: 4009_5000h

LP_FLEXCOMM4 base address: 400B_4000h

LP_FLEXCOMM5 base address: 400B_5000h

LP_FLEXCOMM6 base address: 400B_6000h

LP_FLEXCOMM7 base address: 400B_7000h

LP_FLEXCOMM8 base address: 400B_8000h

LP_FLEXCOMM9 base address: 400B_9000h

Offset	Register	Width (In bits)	Access	Reset value
FF4h	Interrupt Status (ISTAT)	32	R	0000_0000h
FF8h	Peripheral Select and ID (PSELID)	32	RW	0010_3070h

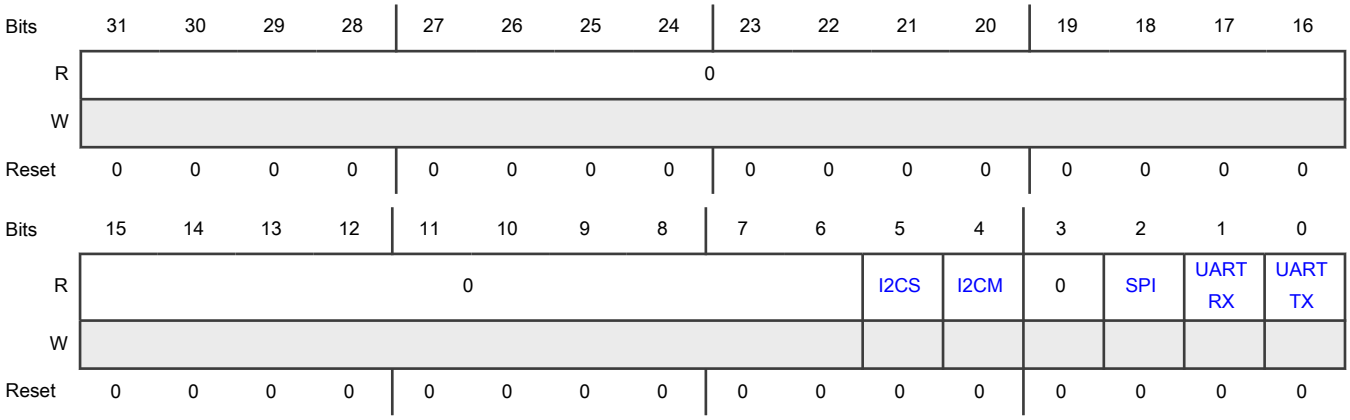
66.2.6.1.2 Interrupt Status (ISTAT)

Offset

Register	Offset
ISTAT	FF4h

Function
Identifies the source of the LP_FLEXCOMM interface single interrupt.

Diagram



Fields

Field	Function
31-6 —	Reserved
5 I2CS	I2C Subordinate Interrupt Indicates if the I2C subordinate has asserted (set) an interrupt or not (remains clear). 0b - Clear 1b - Set
4 I2CM	I2C Controller Interrupt Indicates if the I2C controller has asserted (set) an interrupt or not (remains clear).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Clear 1b - Set
3 —	Reserved
2 SPI	SPI Interrupt Indicates if the SPI has asserted (set) an interrupt or not (remains clear). 0b - Clear 1b - Set
1 UARTRX	UART RX Interrupt Indicates if the UART receiver has asserted (set) an interrupt or not (remains clear). 0b - Clear 1b - Set
0 UARTTX	UART TX Interrupt Indicates if the UART transmitter has asserted (set) an interrupt or not (remains clear). 0b - Clear 1b - Set

66.2.6.1.3 Peripheral Select and ID (PSELID)

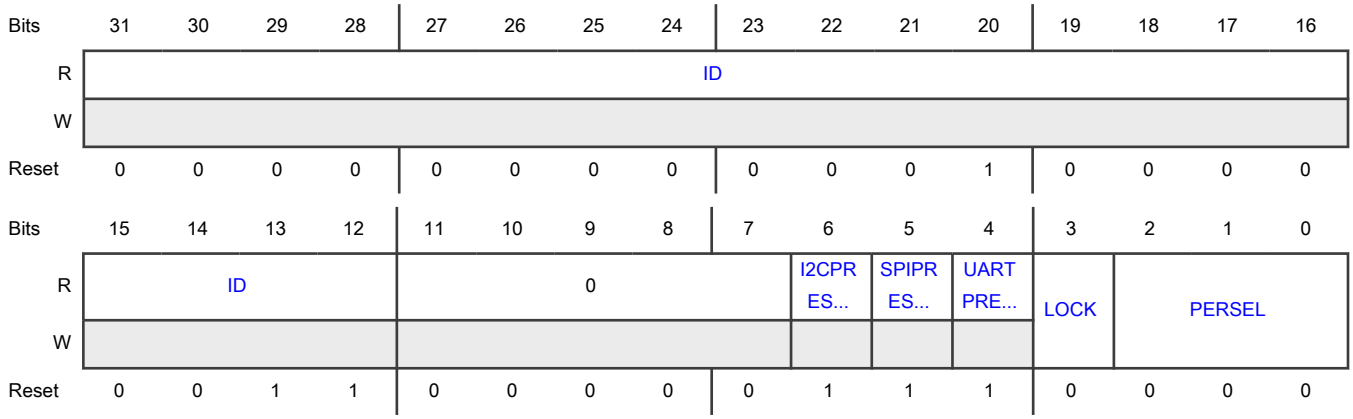
Offset

Register	Offset
PSELID	FF8h

Function

Shows which communications functions are available on a given LP_FLEXCOMM interface and selects the function to be used. It also contains the specific LP_FLEXCOMM interface ID.

Diagram



Fields

Field	Function
31-12 ID	LP_FLEXCOMM interface ID Returns the LP_FLEXCOMM interface ID.
11-7 —	Reserved
6 I2CPRESENT	I2C Present Indicates if the I2C function is supported in a given LP_FLEXCOMM interface. 0b - Not supported 1b - Supported
5 SPIPRESENT	SPI Present Indicates if the SPI function is supported in a given LP_FLEXCOMM interface. 0b - Not supported 1b - Supported
4 UARTPRESENT	UART Present Indicates if the UART function is supported in a given LP_FLEXCOMM interface. 0b - Not supported 1b - Supported
3 LOCK	Lock Locks the Peripheral Select (PERSEL) field until the next reset. 0b - PERSEL is writable 1b - PERSEL is not writable
2-0	Peripheral Select

Table continues on the next page...

Table continued from the previous page...

Field	Function
PERSEL	Selects the peripheral communications function to be used. 000b - No peripheral selected 001b - UART 010b - SPI 011b - I2C 111b - UART and I2C

66.3 Low-Power Inter-Integrated Circuit (LPI2C)

66.3.1 Overview

LPI2C supports an efficient interface to an I2C bus as a controller and target:

- Implements logic support for Standard, Fast, Fast+, HS-mode (target only) and Ultra-Fast modes of operation
- Uses little CPU overhead, with DMA offloading of FIFO register accesses

LPI2C also complies with the System Management Bus (SMBus) Specification, version 3. The SMBus is a single-ended simple two-wire bus, which is typically used for low-bandwidth communications.

The Inter-Integrated Circuit (I^2C) serial bus is multi-controller, multi-target, packet-switched, and single-ended, and is often used to attach microcontroller ICs to lower-speed peripheral ICs.

NOTE

Terminology in this chapter has been updated to align with I^2C-bus specification, Rev. 7.0, as shown in Table 550.

Table 550. Updated terms

Updated term	Deprecated term
Controller	Master
Target	Slave

66.3.1.1 Block diagram

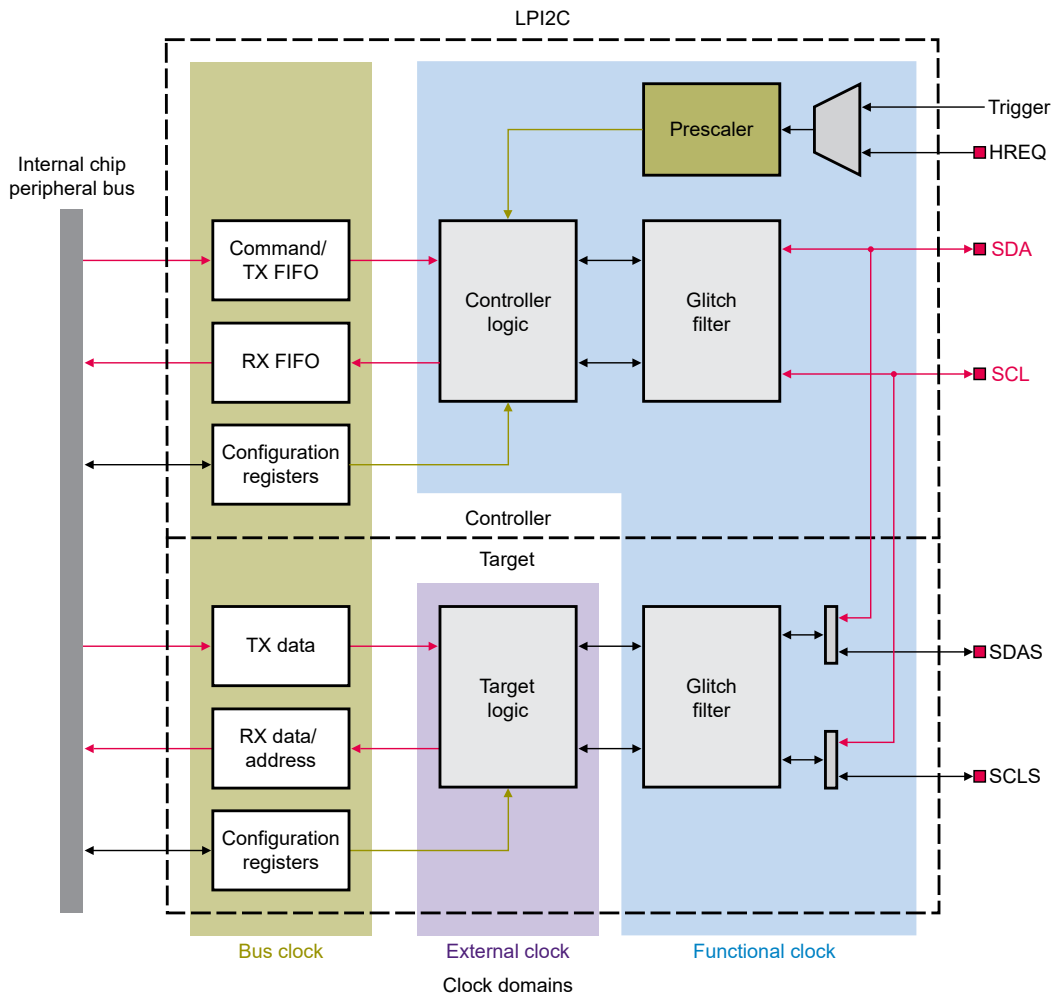


Figure 375. Block diagram

66.3.1.2 Features

LPI2C supports:

- Standard, Fast, Fast+ and Ultra Fast modes
- HS mode in target mode
- Multicontroller, including synchronization and arbitration, means that any number of controller nodes can be present. Also, controller and target roles can be changed between messages (after a Stop signal is sent).
- Clock stretching. Used on the SCL line, as an I2C flow control mechanism.
- Arbitration for when the system has more than one controller. When used on the SDA line, ensures that there is only one I2C transmitter at a time.
- General call, seven-bit addressing, and ten-bit addressing
- Software reset, Start byte, and device ID (also require software support)

The LPI2C controller supports:

- Command and transmit FIFO of 8 words (8-bit transmit data + 3-bit command)

- Receive FIFO of 8 words (8-bit receive data).
- Command FIFO waiting for an I2C idle bus before initiating a transfer
- Initiation of repeated Start and Stop conditions and one or more controller-receiver transfers by command FIFO
- Stop condition generation from command FIFO, or automatic generation of Stop condition when the transmit FIFO is empty
- Host request input to control the start time of an I2C bus transfer
- Interrupt generation on data match and unwanted data rejection, via flexible receive data match
- Flags and optional interrupt signals at repeated Start condition, Stop condition, loss of arbitration, unexpected NACK, and command word errors
- Configurable bus idle timeout and pin-stuck-low timeout

The LPI2C target supports:

- Separate I2C target registers to minimize software overhead because of controller or target switching
- 7-bit or 10-bit addressing, address range, SMBus alert, and general call address.
- Transmit data register that supports interrupt or DMA requests
- Receive data register that supports interrupt or DMA requests
- Software-controllable ACK or NACK, with optional clock stretching on ACK or NACK field
- Configurable clock stretching, to avoid transmit-FIFO-underrun and receive-FIFO-overrun errors
- Flags and optional interrupt at end of packet, Stop condition, or bit error detection

66.3.2 Functional description

66.3.2.1 Controller mode

The LPI2C controller logic operates independently from the target logic to perform all controller-mode transfers on the I2C bus.

66.3.2.1.1 Transmit and Command FIFO commands

The transmit FIFO stores command data to initiate various I2C operations. The following operations can be initiated through commands in the transmit FIFO:

- Start or repeated Start condition with address byte, expecting ACK or NACK.
- Transmit data. This operation is the default for zero-extended-byte writes to the transmit FIFO.
- Receive 1-256 bytes of data. You can configure this operation to discard received data and not to store it in the receive FIFO.
- Stop condition. You can configure this operation to send a Stop condition when the transmit FIFO is empty.

Multiple transmit and receive commands can be inserted between the Start and Stop conditions. To comply with the I2C specification, transmit and receive commands must not be interleaved. The receive data command and the receive data and discard commands can be interleaved. This interleaving ensures that only the desired received data is stored in the receive FIFO (or compared with the data match logic).

The LPI2C controller automatically transmits a NACK on the last byte of a receive data command. It transmits the NACK unless the next command in the FIFO is also a receive data command. If the transmit FIFO is empty when a receive data command completes, a NACK is also automatically transmitted.

The LPI2C controller supports 10-bit addressing via a (repeated) Start condition, followed by a transmit data byte containing the second address byte, followed by any number of data bytes with the controller transmit data.

A Start or repeated Start condition expecting a NACK (for example, HS mode controller code) must be followed by a Stop or (repeated) Start condition.

66.3.2.1.2 Controller operations

When LPI2C is enabled, it monitors the I2C bus to detect when the I2C is idle (MSR[BBF]). If either SCL or SDA are low, the I2C bus is no longer considered idle. The bus becomes idle if a Stop condition is detected or if a bus idle timeout is detected (as configured by MCFGR2[BUSIDLE]). After the bus is idle, if the transmit FIFO is not empty and the host request is asserted or disabled, the LPI2C controller initiates a transfer on the bus. This transfer involves the following steps:

1. Wait the bus idle time equal to (MCCR0[CLKLO] + 1) multiplied by the prescaler (MCFGR1[PRESCALE]).
2. Transmit a Start condition and address byte using the timing configuration in Controller Clock Configuration 0 (MCCR0). If an HS mode transfer is configured, the timing configuration from Controller Clock Configuration 1 (MCCR1) is used instead.
3. Perform controller transmit or controller receive transfers, as configured by the transmit FIFO.
4. Transmit NACK on the last byte of a controller receive transfer. This action is performed unless the next command in the transmit FIFO is also a receive data command and the transmit FIFO is not empty.
5. Transmit a repeated Start or Stop condition as configured by the transmit FIFO or MCFGR1[AUTOSTOP]. A repeated Start can change which timing configuration register is used.

When MCFGR0[RELAX] is 1, the LPI2C controller does not wait for the I2C bus to be idle before starting a transfer. It also relaxes some restrictions on the order of FIFO commands. For example, it allows a Stop command when it is idle to generate a Start condition, followed by one SCL clock pulse, and then a Stop condition.

When the LPI2C controller is disabled, LPI2C continues emptying the transmit FIFO until a Stop condition is transmitted. (The controller could be disabled due to MCR[MEN] being 0, or automatically due to mode entry.) However, LPI2C no longer stalls the I2C bus by waiting for the transmit or receive FIFO. After the transmit FIFO is empty, LPI2C generates a Stop condition automatically.

The LPI2C controller can stall the I2C bus under certain conditions. This stalling results in SCL pulled low continuously on the first bit of a byte, until these conditions change:

- The LPI2C controller is enabled and busy, the transmit FIFO is empty, and MCFGR1[AUTOSTOP] is 0. The LPI2C controller continues to stall the bus until the transmit FIFO is loaded with more data.
- The LPI2C controller is enabled and receiving data, receive data is not being discarded (due to command or receive data match), and the receive FIFO is full. The LPI2C controller continues to stall the I2C bus until the receive FIFO is emptied.

The LPI2C controller aborts the existing transfer when MCFGR0[ABORT] becomes 1. This action causes the LPI2C controller to complete the existing word and then transmit a Stop condition. If the receive FIFO is full when the receive operation is aborted, the last received data is discarded. A new transfer does not start while MCFGR0[ABORT] is 1.

66.3.2.1.3 Receive FIFO and data matching

The receive FIFO stores receive data during controller-receiver transfers. You can configure the LPI2C controller to discard received data instead of storing it in the receive FIFO. This option is configured via the command word in the transmit FIFO.

Received data supports a receive data match function that can match received data against one of two bytes, or against a masked data byte. You can configure the data match function to compare only the first one or two data words received since the last (repeated) Start condition. Received data that is already discarded due to the command word cannot cause a data match. It delays the match on the first data word received until after the discarded data is received.

You can configure the receiver match function to discard all received data until a data match is detected, using MCFGR0[RDMO]. Following a data match, write 0 to MCFGR0[RDMO] before writing 0 to MSR[DMF] to allow all subsequent data to be received.

66.3.2.1.4 Timing parameters

The LPI2C controller can configure the following timing parameters. Parameters are configured separately for HS mode (Controller Clock Configuration 1 (MCCR1)) and other modes (Controller Clock Configuration 0 (MCCR0)). This separation allows the HS mode controller code to be sent using regular timing parameters. Then it allows a switch to HS mode timing (following a repeated Start) until the next STOP condition.

Configure the LPI2C controller timing parameters, measured in LPI2C functional clock cycles, as shown in Table 551. You must configure these parameters to meet the I2C timing specification for the required mode.

Table 551. Timing parameters

I2C specification timing parameter	I2C specification timing symbol	LPI2C timing parameter (in LPI2C functional clock cycles)
SCL clock period	tSCL	(CLKHI + CLKLO + 2 + SCL_LATENCY) × (2 ^ PRESCALE)
Hold time (repeated) Start condition	tHD:STA	(SETHOLD +1) × (2 ^ PRESCALE)
Low period of the SCL clock	tLOW	(CLKLO + 1) × (2 ^ PRESCALE)
High period of the SCL clock	tHIGH	(CLKHI + 1 + SCL_LATENCY) × (2 ^ PRESCALE)
Setup time for a repeated Start condition or Stop condition	tSU:STA, tSU:STO	(SETHOLD + 1 + SCL_LATENCY) × (2 ^ PRESCALE)
Data hold time	tHD:DAT	(DATAVD + 1) × (2 ^ PRESCALE)
Data setup time	tSU:DAT	(SDA_LATENCY + 1) × (2 ^ PRESCALE)
Bus free time between a Stop and Start condition	tBUF	(CLKLO + 1 + SDA_LATENCY) × (2 ^ PRESCALE)
Data valid time, data valid acknowledge time	tVD:DAT, tVD:ACK	(DATAVD + 1) × (2 ^ PRESCALE)

Table 552 defines the latency parameters. These parameters assume that the risetime is less than one LPI2C functional clock cycle. The risetime depends on a number of factors, including the I/O propagation delay, the I2C bus loading, and the external pullup resistor sizing. A larger risetime increases the number of cycles that the signal takes to propagate through the synchronizer (and glitch filter), which increases the latency.

Table 552. Synchronization latency

Timing parameter	Timing definition
SCL_LATENCY	ROUNDDOWN ((2 + FILTSCL + SCL_RISETIME) ÷ (2 ^ PRESCALE))
SDA_LATENCY	ROUNDDOWN ((2 + FILTSDA + SDA_RISETIME) ÷ (2 ^ PRESCALE))

The following timing restrictions must be enforced to avoid unexpected Start or Stop conditions on the I2C bus. These restrictions also avoid unexpected Start or Stop conditions detected by the LPI2C controller. The timing restrictions can be summarized as "SDA cannot change when SCL is high outside a transmitted (repeated) Start or Stop condition."

Table 553. LPI2C timing parameter restrictions

Timing parameter	Minimum	Maximum	Comment
CLKLO	03h	—	CLKLO x (2 ^ PRESCALE) > SCL_LATENCY
CLKHI	01h	—	Configure CLKHI to meet the duty cycle requirements in the I2C specification
SETHOLD	02h	—	SETHOLD × (2 ^ PRESCALE) > SDA_LATENCY

Table continues on the next page...

Table 553. LPI2C timing parameter restrictions (continued)

Timing parameter	Minimum	Maximum	Comment
DATAVD	01h	CLKLO – SDA_LATENCY – 1	Configure DATAVD to meet the data hold requirement in the I2C specification
FILTSCL	00h	[CLKLO × (2 ^ PRESCALE)] – 3	FILTSCL and FILTSDA are the only parameters not multiplied by (2 ^ PRESCALE)
FILTSDA	FILTSCL	[CLKLO × (2 ^ PRESCALE)] – 3	Configuring FILTSDA greater than FILTSCL can delay the SDA input to compensate for board level skew
BUSIDLE	(CLKLO + SETHOLD + 2) × 2	—	Must also be greater than (CLKHI + 1)

See the UM10204, *I2C-bus specification and user manual*.

See Application information for example LPI2C timing configurations.

66.3.2.1.5 Error conditions

The LPI2C controller monitors errors while it is active. The following conditions generate an error flag and block a new Start condition from being sent, until the flag is cleared by software:

- A Start or Stop condition is detected and is not generated by the LPI2C controller (MSR[ALF] becomes 1).
- Transmitting data on SDA and different values are received (MSR[ALF] becomes 1). LPI2C controller stops driving SDA immediately, but continues to drive SCL for the remainder of the byte.
- NACK is detected when transmitting data, and MCFGR1[IGNACK] is 0 (MSR[NDF] becomes 1).
- NACK is detected and is expecting ACK for the address byte, and MCFGR1[IGNACK] is 0 (MSR[NDF] becomes 1).
- ACK is detected and is expecting NACK for the address byte, and MCFGR1[IGNACK] is 0 (MSR[NDF] becomes 1).
- Transmit FIFO is requesting to transmit or receive data without a Start condition (MSR[FEF] becomes 1). This restriction is ignored when MCFGR0[RELAX] is 1.
- SCL (or SDA if MCFGR1[TIMECFG] is 1) is low for (MCFGR2[TIMELOW] × 256) prescaler cycles without a pin transition (MSR[PLTF] becomes 1).

You must respond to MSR[PLTF] to terminate the existing command. You can terminate the command cleanly by writing 0 to MCR[MEN], or you can terminate it abruptly by writing 1 to MCR[RST].

You can use MCFGR0[RELAX] to attempt to recover a target with SDA stuck low. If MCFGR0[RELAX] is 1, the LPI2C controller does not wait for the I2C bus to be idle before starting a transfer. Initiating a Start command with address FFh, then the Stop condition, should generate sufficient SCL clock edges to cause the target to release SDA.

You can use MCFGR2[BUSIDLE] to force the I2C bus to be considered idle when SCL and SDA remain high for (BUSIDLE + 1) prescaler cycles. The bus is considered idle when the LPI2C controller is first enabled. When BUSIDLE is configured greater than zero, then SCL or SDA must be high for (BUSIDLE + 1) prescaler cycles before the I2C bus is considered idle.

66.3.2.1.6 DMA support

For efficient DMA transfers to the transmit and command FIFO, two alias regions support incrementing 8-bit, 16-bit, or 32-bit write accesses to that FIFO.

- Controller Transmit Command Burst (MTCBR0 - MTCBR127) is a 512-byte region that supports pushing CMD and DATA into the transmit FIFO.

- Transmit Data Burst (MTDBR0 - MTDBR252) is a 1024-byte region that supports pushing zero extended DATA into the transmit FIFO.

The two regions are contiguous. A DMA transfer can start in the MTCBR region to initialize the transfer including START condition with address. The same transfer can complete in the MTDBR region with the data to transmit without changing the transfer size.

The transmit FIFO prevents writes that overflow the FIFO, but this prevention does not signal an error. Do not perform 32-bit writes to the MTDBR unless there are four empty slots in the transmit FIFO. Do not perform 16-bit writes to the MTDBR unless there are two empty slots in the transmit FIFO.

66.3.2.1.7 Pin configuration

Configuration	Description
Open-drain support	The LPI2C controller defaults to open-drain configuration of the SDA and SCL pins. Support for true open drain depends on the specific device, and requires the pins where LPI2C pins are muxed to support true open drain.
HS mode support	Support for HS mode depends on the specific device. This mode requires the SCL pin to support the current source pullup required in the I2C specification.
Ultra-Fast mode support	The LPI2C controller supports the output-only push-pull function required for I2C Ultra-Fast mode using the SDA and SCL pins. Support for Ultra-Fast mode also requires MCFGR1[IGNACK] to be 1.
Push-pull two-wire support	A push-pull two-wire configuration is available to the LPI2C controller. If LPI2C is the only controller and all I2C pins on the bus are at the same voltage, this configuration may support a partial HS mode. A partial HS mode, not a full HS mode, because this configuration actively drives high rather than enabling a current service pull-up. This configuration sets the SCL pin as push-pull for every clock except the first clock pulse, to allow HS-mode-compatible targets to perform clock stretching. In this mode, the SDA pin is tristated for controller-receive data bits and controller-transmit ACK/NACK bits, and is configured as push-pull at other times. To avoid the risk of contention when SDA is push-pull, configure the pin for open-drain operation, as part of the device-specific configuration.
Push-pull four-wire support	The push-pull four-wire configuration separates the SCL input data and output data into separate pins. It also separates the SDA input data and output data into separate pins. The SCL/SDA pins are used for input data; the SCLS/SDAS pins are used for output data, with configurable polarity. This configuration simplifies external connections when connecting the LPI2C to the I2C bus through external level shifters or discrete components. When using this four-wire configuration, the LPI2C controller logic and LPI2C target logic cannot connect to separate I2C buses.

66.3.2.2 Target mode

To perform all target mode transfers on the I2C bus, the LPI2C target logic operates independently from the LPI2C controller logic.

66.3.2.2.1 Address matching

You can configure the LPI2C target:

- To match one of two addresses, using either 7-bit or 10-bit addressing modes for each address.
- To match a range of addresses in either 7-bit or 10-bit addressing modes.
- To match the general call address and generate appropriate flags.
- To match the SMBus alert address and generate appropriate flags.
- To detect the HS mode controller code address, and to disable the digital filters and output valid delay time until the next Stop condition is detected.

After a valid address is matched, the LPI2C target automatically performs target-transmit or target-receive transfers until:

- A NACK is detected (unless SCFGR1[IGNACK] becomes 1).
- A bit error is detected (the LPI2C target is driving SDA, but a different value is sampled).
- A (repeated) Start or Stop condition is detected.

When SCFGR1[RXALL] is 1, the LPI2C target receives all addresses and data on the I2C bus. Unless the address is matched, it never drives the SDA pin. Configure the LPI2C target to match only on 7-bit address mode when this feature is enabled. Software can support 10-bit address mode. This option is intended for debugging the contents of an I2C transfer between another controller and target. Likewise, SCFGR1[SDCFG] and SCFGR1[RSCFG] configure the Stop or repeated Start flags to assert on all Stop or repeated Start conditions, even when there is no address match.

66.3.2.2.2 Transmit and receive data

Target Transmit Data (STDR) and Target Receive Data (SRDR) are double-buffered and only update during a target-transmit and target-receive transfer, respectively.

You can configure the target address that was received to be read from SRDR (for example, when using DMA to transfer data) or from Target Address Status (SASR).

You can configure STDR to request data only after a target-transmit transfer is detected. You can also configure it to request new data whenever STDR is empty.

Write to STDR only when SSR[TDF] is set.

Read SRDR only when SSR[RDF] is set, or when SSR[AVF] is set and SCFGR1[RXCFG] = 1.

Read SASR only when SSR[AVF] is set.

66.3.2.2.3 Read request

The LPI2C target generates a read request when the I2C bus is idle and you write 1 to SCFGR0[RDREQ]. This occurrence causes the LPI2C target to pull the SDA pin low until either the first SCL falling edge or SCFGR0[RDREQ] becomes 0. Following the next Stop condition or a software timeout, you can proceed as follows:

- If the LPI2C target is accessed at the correct address, the read request is acknowledged, and you must write 0 to SCFGR0[RDREQ].
- If SCFGR0[RDACK] is 1, the read request is acknowledged by a Start followed by one SCL pulse followed by a STOP condition. You must write 0 to SCFGR0[RDREQ].
- If neither of the first two cases occurred, the read request is not acknowledged. You must write 0 and then 1 to SCFGR0[RDREQ] after an appropriate delay, provided the I2C bus is idle.
- If a software timeout occurs, the read request is not acknowledged, and you should write 0 to SCFGR0[RDREQ]. Another request can be generated after an appropriate delay.

Writing to SCFGR0[RDREQ] when the I2C bus is busy results in a logic 0 being written. Write 1 to SCFGR0[RDREQ], then confirm that the register is written to correctly. If the register does not update correctly, the I2C bus is no longer idle and the read request must be attempted later.

66.3.2.2.4 Clock stretching

The LPI2C target supports many configurable options for clock stretching. You can configure these conditions to perform clock stretching:

- SSR[AVF] is set during the ninth clock pulse of the address byte.
- SSR[TDF] is set during the ninth clock pulse of a target-transmit transfer.
- SSR[RDF] is set during the ninth clock pulse of a target-receive transfer.
- SSR[TAF] is set during the eighth clock pulse of an address byte or a target-receive transfer. In HS mode, this option is disabled.

- Clock stretching can be extended for a number of cycles equal to the value of SCFGR2[CLKHOLD] cycles. This stretching allows additional setup time to sample the SDA pin externally. In HS mode, this option is disabled.

When clock stretching is enabled, clock stretching extends for one peripheral bus clock cycle after SDA updates, unless extended by the SCFGR2[CLKHOLD] configuration.

66.3.2.2.5 Timing parameters

The LPI2C target can configure the following timing parameters:

- SDA data valid time from SCL negation to SDA update
- SCL hold time when clock stretching is enabled to increase setup time when sampling SDA externally
- SCL glitch filter time
- SDA glitch filter time

These parameters are disabled when SCR[FILTEN] is 0, when SCR[FILTDZ] is 1 in Doze mode, and when LPI2C target detects HS mode. When disabled, the LPI2C target is clocked directly from the I2C bus. In this case, the target may not satisfy all timing requirements of the I2C specification (such as SDA minimum hold time in Standard/Fast mode).

The LPI2C target places the following restrictions on the timing parameters:

- You must configure SCFGR2[FILTSDA] to be greater than or equal to SCFGR2[FILTSCL] (unless compensating for board level skew between SDA and SCL).
- You must configure SCFGR2[DATAVD] to be less than the minimum SCL low period.

66.3.2.2.6 Error conditions

The LPI2C target can flag the following error conditions:

- SSR[BEF] is set when the LPI2C target is driving SDA but it samples a different value than what is expected.
- SSR[FEF] is set due to a transmit data underrun or a receive data overrun. To eliminate the possibility of underrun and overrun, enable clock stretching. When SCFGR1[RXNACK] is 1, the LPI2C target generates a NACK on a receive data overrun.
- SSR[FEF] is also set due to an address overrun, but only when SCFGR1[RXCFG] is 1. To eliminate the possibility of overrun, enable clock stretching. When SCFGR1[RXNACK] is 1, the LPI2C target generates a NACK on an address overrun regardless of the value of SCFGR1[RXCFG].

The LPI2C target does not implement a timeout due to SCL or SDA being stuck low. If this detection is required, use the LPI2C controller logic so you can reset the LPI2C target when this condition is detected.

66.3.2.3 Low-power modes

LPI2C remains functional during low-power modes, if MCR[DOZEN] = 0 and LPI2C uses an external or internal clock source that remains enabled. LPI2C can generate an interrupt or DMA request to cause a wake-up from low-power modes.

You can configure LPI2C to be disabled in low-power modes when MCR[DOZEN] = 1. In this case, LPI2C waits for the current transfer to complete any pending operation.

NOTE

See the chip-specific information for low-power modes available on your chip.

66.3.2.4 Debug mode

Table 554. Debug mode

Mode	LPI2C operation
Debug	If MCR[DBGEN] = 1, can continue operating in Debug mode.

66.3.2.5 Peripheral triggers

The connection of the LPI2C peripheral triggers to other peripherals depends upon the specific device being used.

Table 555. LPI2C triggers

Trigger	Description
Controller output trigger	Generates an output trigger that can be connected to other peripherals on the device. The controller output trigger asserts on either a repeated Start or a Stop condition. The trigger remains asserted for one cycle of the LPI2C functional clock divided by MCFGR1[PRESCALE].
Target output trigger	Generates an output trigger that can be connected to other peripherals on the device. The target output trigger asserts on either a repeated Start or a Stop condition that occurs after a target address match. The target output trigger remains asserted until the next target SCL pin negation.
Input trigger	To control the start of a LPI2C bus transfer, the LPI2C input trigger can be selected instead of the HREQ input. The input trigger is synchronized. To be detected, the input trigger must assert for at least two cycles of the LPI2C functional clock divided by the value of MCFGR1[PRESCALE]. When LPI2C is busy, the HREQ input (and therefore the input trigger) is ignored.

66.3.2.6 Clocking

Table 556. LPI2C clocks

Clock	Description
LPI2C functional clock	The LPI2C functional clock is asynchronous to the bus clock. It can remain enabled in low-power modes to support I2C bus transfers by the LPI2C controller. The functional clock is also used by the LPI2C target to support digital filter and data hold time configurations. The LPI2C controller divides the functional clock by a prescaler (MCFGR1[PRESCALE]) and the resulting frequency must be at least eight times faster than the I2C bus bandwidth.
External clock	The LPI2C target logic is clocked directly from the external pins. These pins are SCL and SDA, or SCLS and SDAS if the controller and target are implemented on separate pins). This clocking allows the LPI2C target to remain operational, even when the LPI2C functional clock is disabled. **NOTE** If the LPI2C functional clock is disabled, the LPI2C target digital filter must be disabled. This condition can affect compliance with some timing parameters of the I2C specification, such as data hold time.
Bus clock	The bus clock is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPI2C controller and target registers.

For chip-specific clocking information, see the Clocking chapter.

66.3.2.7 Reset

Table 557. LPI2C resets

Reset	Description
Chip reset	The logic and registers for the LPI2C controller and target are reset to their default states after a chip reset.
Software reset	The LPI2C controller implements a software reset field in its control register. MCR[RST] resets all controller logic and registers to their default states, except for Controller Control (MCR) itself. The LPI2C target implements a software reset field in its control register. SCR[RST] resets all target logic and registers to their default states, except for Target Control (SCR) itself.
FIFO reset	The LPI2C controller implements write-only control fields that reset the transmit FIFO (MCR[RTF]) and receive FIFO (MCR[RRF]). After a FIFO is reset, that FIFO is empty. The LPI2C target implements write-only control fields that reset the transmit data register (SCR[RTF]) and receive data register (SCR[RRF]). After a data register is reset, that data register is empty.

66.3.2.8 Interrupts and DMA requests

Depending on the configuration, interrupts and DMA requests can be combined:

- LPI2C controller and target interrupts
- LPI2C controller and target transmit DMA requests
- LPI2C controller and target receive DMA requests

66.3.2.8.1 Controller mode

Table 558 lists the Controller mode sources that can generate LPI2C controller interrupts and LPI2C controller transmit and receive DMA requests.

Table 558. Controller interrupts and DMA requests

Status flag	Description	Can generate		
		Interrupt?	DMA request?	Low-power wake-up?
Transmit Data Flag (MSR[TDF])	Data can be written to transmit FIFO, as configured by MFCR[TXWATER].	Y	TX	Y
Receive Data Flag (MSR[RDF])	Data can be read from the receive FIFO, as configured by MFCR[RXWATER].	Y	RX	Y
End Packet Flag (MSR[EPF])	Controller has transmitted a repeated Start or Stop condition.	Y	N	Y
Stop Detect Flag (MSR[SDF])	Controller has transmitted a Stop condition (and the LPI2C controller is idle, if MCFGR1[STOPCFG] = 1) .	Y	N	Y
NACK Detect Flag (MSR[NDF])	During an address byte, the controller expects an ACK but detects a NACK. During an address byte, the controller expects a NACK but detects an ACK.	Y	N	Y

Table continues on the next page...

Table 558. Controller interrupts and DMA requests (continued)

Status flag	Description	Can generate		
		Interrupt?	DMA request?	Low-power wake-up?
	During a controller-transmitter data byte, the controller detects a NACK.			
Arbitration Lost Flag (MSR[ALF])	The controller lost arbitration due to a Start or Stop condition detected at the wrong time, or the controller was transmitting data but received data different from the data that was transmitted.	Y	N	Y
FIFO Error Flag (MSR[FEF])	The controller expects a Start condition in the command FIFO, but the next entry in the command FIFO is not a Start condition.	Y	N	Y
Pin Low Timeout Flag (MSR[PLTF])	Pin low timeout is enabled and SCL (or SDA, if configured) is low for longer than the configured timeout.	Y	N	Y
Data Match Flag (MSR[DMF])	The received data matches the configured data match, but the received data is not discarded due to a command FIFO entry.	Y	N	Y
Start Detect Flag (MSR[STF])	A Start condition is detected on the I2C bus (and the LPI2C controller is idle, if MCFGR1[STARTCFG] = 1).	Y	N	Y
Controller Busy Flag (MSR[MBF])	LPI2C controller is busy transmitting or receiving data.	N	N	N
Bus Busy Flag (MSR[BBF])	LPI2C controller is enabled and activity is detected on the I2C bus, but no Stop condition is detected and no bus idle timeout (if enabled) occurred.	N	N	N

66.3.2.8.2 Target mode

Table 559 lists the target mode sources that can generate LPI2C target interrupts and LPI2C target transmit and receive DMA requests.

Table 559. Target interrupts and DMA requests

Status flag	Description	Can generate		
		Interrupt?	DMA request?	Low-power wake-up?
Transmit Data Flag (SSR[TDF])	Data can be written to Target Transmit Data (STDR).	Y	TX	Y
Receive Data Flag (SSR[RDF])	Data can be read from Target Receive Data (SRDR).	Y	RX	Y

Table continues on the next page...

Table 559. Target interrupts and DMA requests (continued)

Status flag	Description	Can generate		
		Interrupt?	DMA request?	Low-power wake-up?
Address Valid Flag (SSR[AVF])	Address can be read from Target Address Status (SASR).	Y	RX	Y
Transmit ACK Flag (SSR[TAF])	ACK or NACK can be written to Target Transmit ACK (STAR).	Y	N	Y
Repeated Start Flag (SSR[RSF])	Target has detected an address match followed by a repeated Start condition.	Y	RX	Y
Stop Detect Flag (SSR[SDF])	Target has detected an address match followed by a Stop condition.	Y	RX	Y
Bit Error Flag (SSR[BEF])	Target was transmitting data, but received data is different from what was transmitted.	Y	N	Y
FIFO Error Flag (SSR[FEF])	This flag is set by: • Transmit data underrun • Receive data overrun • Address status overrun when SCFGR1[RXCFG] = 1 This flag can only be set when clock stretching is disabled.	Y	N	Y
Address Match 0 Flag (SSR[AM0F])	Target detected an address match SAMR[ADDR0].	Y	N	N
Address Match 1 Flag (SSR[AM1F])	Target detected an address match with SAMR[ADDR1] or using an address range.	Y	N	N
General Call Flag (SSR[GCF])	Target detected an address match with the general call address.	Y	N	N
SMBus Alert Response Flag (SSR[SARF])	Target detected an address match with the SMBus alert address.	Y	N	N
Target Busy Flag (SSR[SBF])	LPI2C target is busy receiving an address byte or is transmitting or receiving data.	N	N	N
Bus Busy Flag (SSR[BBF])	LPI2C target is enabled and a Start condition is detected on I2C bus, but no Stop condition detected.	N	N	N

66.3.2.8.3 End-of-packet DMA transfer

End-of-packet functionality serves serial interfaces where the transfer size may not be known in advance and the data is pushed by an external device. Examples include UART receive, I2C Target mode, and SPI Target mode. End-of-packet processing is intended to ensure that data does not become stranded in either the receive FIFO or DMA receive buffer. Support for end-of-packet processing must be implemented in both the serial interfaces and DMA controller.

The condition that signals the end of packet differs for each serial interface but is processed by the serial peripheral and DMA in the same way. For example:

- UART end of packet is signaled by an idle line condition.

- I2C end of packet is signaled by Stop or repeated Start condition.
- SPI end of packet is signaled by PCS negation.

When the serial peripheral is configured to signal an end-of-packet condition to the DMA and the serial interface detects an end-of-packet condition, it asserts the DMA request for the receive FIFO regardless of the watermark configuration. For larger watermark configurations, this behavior ensures that the last few words of the transfer are first flushed from the receive FIFO.

The DMA then reads the contents of the receive FIFO, depending on the first word in the FIFO:

- If the receive FIFO is empty, the serial interface signals an end-of-packet condition to the DMA controller.
- If the receive FIFO is not empty, but the first word in the FIFO starts a new packet, data is not pulled from the receive FIFO. Also, the serial interface signals an end-of-packet condition to the DMA controller.
- If the receive FIFO is not empty, and the first word in the FIFO does not start a new packet, the DMA transfers data as normal.

Because the DMA may transfer multiple words per request, the end-of-packet condition persists until the DMA minor loop finishes and no additional data is pulled from the FIFO. The status flag that triggered the end-of-packet condition is cleared when the minor loop completes after the end of packet is signaled to the DMA controller.

When the DMA detects the end-of-packet condition, it writes all received words up to the end of packet into system memory. It also saves the destination address for the word after the last valid data. The DMA then terminates the channel as if the major loop completed, including final offsets and optional interrupts, channel linking, and scatter-gather. The final destination address can be saved to system memory or is available in the destination address register.

Because the DMA terminates the major loop, the receive FIFO is not serviced until the DMA is reconfigured. You can perform this reconfiguration through software or hardware (for example, channel linking or scatter-gather). Minimize this delay to avoid receiver FIFO overrun. Automatic DMA end-of-packet processing is not recommended when there are only a few words transferred between end-of-packet conditions. In this case, the DMA spends more time processing the end of packet than transferring the data. For example, to avoid an excessive number of idle conditions, increase the UART idle line length as needed.

66.3.3 External signals

Table 560. External signals

Signal	Name	Two-wire scheme	Four-wire scheme	Direction
SCL	LPI2C clock line	SCL	In Four-Wire mode, this pin is the SCL input pin.	Input or output
SDA	LPI2C data line	SDA	In Four-Wire mode, this pin is the SDA input pin.	Input or output
SCLS	Secondary I2C clock line	Not used	In Four-Wire mode, this pin is the SCLS output pin. If LPI2C controller/target are configured to use separate pins, then this pin is the LPI2C target SCL pin.	Input or output
SDAS	Secondary I2C data line	Not used	In Four-Wire mode, this pin is the SDAS output pin. If LPI2C controller/target are configured to use separate pins, then this pin is the LPI2C target SDA pin.	Input or output
HREQ	Host request	When configured for controller (input), if host request is asserted and the I2C bus is idle, it initiates a transfer. When configured for target (output), it asserts while valid data is present in Target Transmit Data (STDR).		Input or output

Table continues on the next page...

Table 560. External signals (continued)

Signal	Name	Two-wire scheme	Four-wire scheme	Direction
		HREQ is an additional pin separate from the two-wire or four-wire scheme.		

Figure 376 shows the two-signal connection.

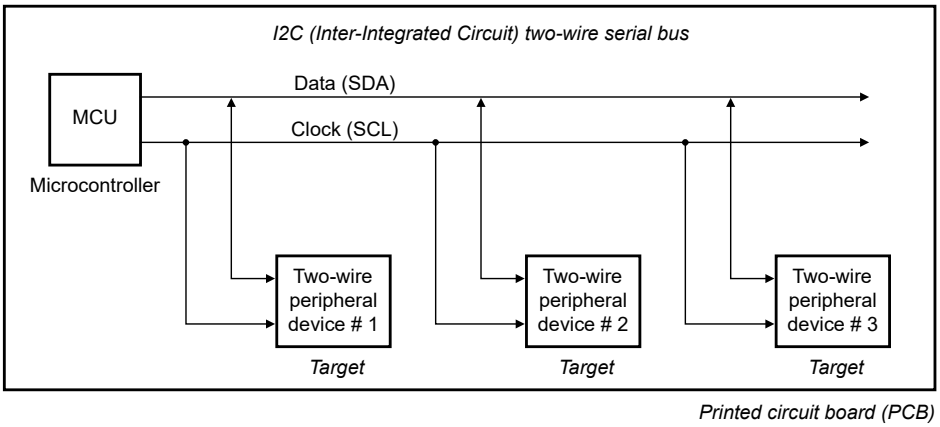


Figure 376. I²C two-wire serial bus

Figure 377 shows a possible four-signal connection.

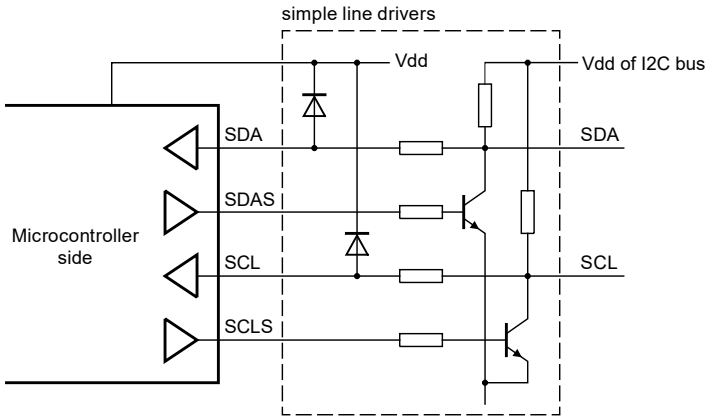


Figure 377. I²C four-wire serial bus

66.3.4 Initialization

To initialize the LPI2C controller:

1. Configure Controller Configuration 0 (MCFGR0)–Controller Configuration 3 (MCFGR3) as required by the application.
2. Configure Controller Clock Configuration 0 (MCCR0) and Controller Clock Configuration 1 (MCCR1) to satisfy the timing requirements of the I2C mode supported by the application.
3. Enable controller interrupts and DMA requests as required by the application.
4. Enable the LPI2C controller by writing 1 to MCR[MEN].

To initialize the LPI2C target:

1. Configure Target Address Match (SAMR) with the I2C address of the target location on the I2C bus.
2. Configure Target Configuration 0 (SCFGR0) and Target Configuration 1 (SCFGR1) as required by the application.
3. Configure Target Configuration 2 (SCFGR2) to satisfy the timing requirements of the I2C mode supported by the application.
4. Enable target interrupts and DMA requests as required by the application.
5. Enable the LPI2C target by writing 1 to SCR[SEN].

66.3.5 Application information

Configure the I2C timing parameters to meet the requirements of the I2C specification. This configuration depends on the supported mode and LPI2C functional clock frequency. When switching between two modes using different clock configuration registers (for example, Fast mode and HS mode), MCFGR1[PRESCALE] must remain constant between the modes.

Table 561. Example timing configurations

I2C mode	Clock frequency	Baud rate	PRESCALE	FILTSCL / FILTSDA	SETHOLD	CLKLO	CLKHI	DATAVD
Standard	8 MHz	100 kbit/s	0h	0h/0h	24h	28h	24h	02h
Standard	48 MHz	100 kbit/s	2h	1h/1h	37h	3Fh	37h	03h
Standard	60 MHz	100 kbit/s	2h	1h/1h	45h	50h	44h	04h
Fast	8 MHz	400 kbit/s	0h	0h/0h	04h	0Bh	05h	02h
Fast+	8 MHz	1 Mbit/s	0h	0h/0h	02h	03h	01h	01h
Fast	48 MHz	400 kbit/s	0h	1h/1h	1Dh	3Eh	35h	0Fh
Fast	48 MHz	400 kbit/s	2h	1h/1h	07h	11h	0Bh	03h
Fast+	48 MHz	1 Mbit/s	2h	1h/1h	03h	06h	04h	04h
HS	48 MHz	3.2 Mbit/s	0h	0h/0h	07h	08h	03h	01h
Fast	60 MHz	400 kbit/s	1h	2h/2h	11h	28h	1Fh	08h
Fast+	60 MHz	1 Mbit/s	1h	2h/2h	07h	0Fh	0Bh	01h
HS	60 MHz	3.33 Mbit/s	1h	0h/0h	04h	04h	02h	01h

66.3.6 Memory map and registers

66.3.6.1 LPI2C register descriptions

Writing to a read-only register or reading from a write-only register can cause bus errors. This module does not check whether programmed values in the registers are correct; you must ensure that valid programmed values are written to the registers.

66.3.6.1.1 LPI2C memory map

LPI2C0 base address: 4009_2800h

LPI2C1 base address: 4009_3800h

LPI2C2 base address: 4009_4800h

LPI2C3 base address: 4009_5800h

LPI2C4 base address: 400B_4800h

LPI2C5 base address: 400B_5800h

LPI2C6 base address: 400B_6800h

LPI2C7 base address: 400B_7800h

LPI2C8 base address: 400B_8800h

LPI2C9 base address: 400B_9800h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0103_0003h
4h	Parameter (PARAM)	32	R	0000_0303h
10h	Controller Control (MCR)	32	RW	0000_0000h
14h	Controller Status (MSR)	32	RW	0000_0001h
18h	Controller Interrupt Enable (MIER)	32	RW	0000_0000h
1Ch	Controller DMA Enable (MDER)	32	RW	0000_0000h
20h	Controller Configuration 0 (MCFGR0)	32	RW	0000_0000h
24h	Controller Configuration 1 (MCFGR1)	32	RW	0000_0000h
28h	Controller Configuration 2 (MCFGR2)	32	RW	0000_0000h
2Ch	Controller Configuration 3 (MCFGR3)	32	RW	0000_0000h
40h	Controller Data Match (MDMR)	32	RW	0000_0000h
48h	Controller Clock Configuration 0 (MCCR0)	32	RW	0000_0000h
50h	Controller Clock Configuration 1 (MCCR1)	32	RW	0000_0000h
58h	Controller FIFO Control (MFCR)	32	RW	0000_0000h
5Ch	Controller FIFO Status (MFSR)	32	R	0000_0000h
60h	Controller Transmit Data (MTDR)	32	W	0000_0000h
70h	Controller Receive Data (MRDR)	32	R	0000_4000h
78h	Controller Receive Data Read Only (MRDROR)	32	R	0000_4000h
110h	Target Control (SCR)	32	RW	0000_0000h
114h	Target Status (SSR)	32	RW	0000_0000h
118h	Target Interrupt Enable (SIER)	32	RW	0000_0000h
11Ch	Target DMA Enable (SDER)	32	RW	0000_0000h
120h	Target Configuration 0 (SCFGR0)	32	RW	0000_0000h
124h	Target Configuration 1 (SCFGR1)	32	RW	0000_0000h
128h	Target Configuration 2 (SCFGR2)	32	RW	0000_0000h
140h	Target Address Match (SAMR)	32	RW	0000_0000h
150h	Target Address Status (SASR)	32	R	0000_4000h
154h	Target Transmit ACK (STAR)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
160h	Target Transmit Data (STDR)	32	W	0000_0000h
170h	Target Receive Data (SRDR)	32	R	0000_4000h
178h	Target Receive Data Read Only (SRDROR)	32	R	0000_4000h
200h - 3FCh	Controller Transmit Command Burst (MTCBR0 - MTCBR127)	32	W	0000_0000h
400h - 7F0h	Transmit Data Burst (MTDBR0 - MTDBR252)	32	W	0000_0000h

66.3.6.1.2 Version ID (VERID)

Offset

Register	Offset
VERID	0h

Function
Contains version numbers for the module design and feature set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module design specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module design specification.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-0 FEATURE	Feature Specification Number Returns the feature set number. 0000_0000_0000_0010b - Controller only, with standard feature set 0000_0000_0000_0011b - Controller and target, with standard feature set

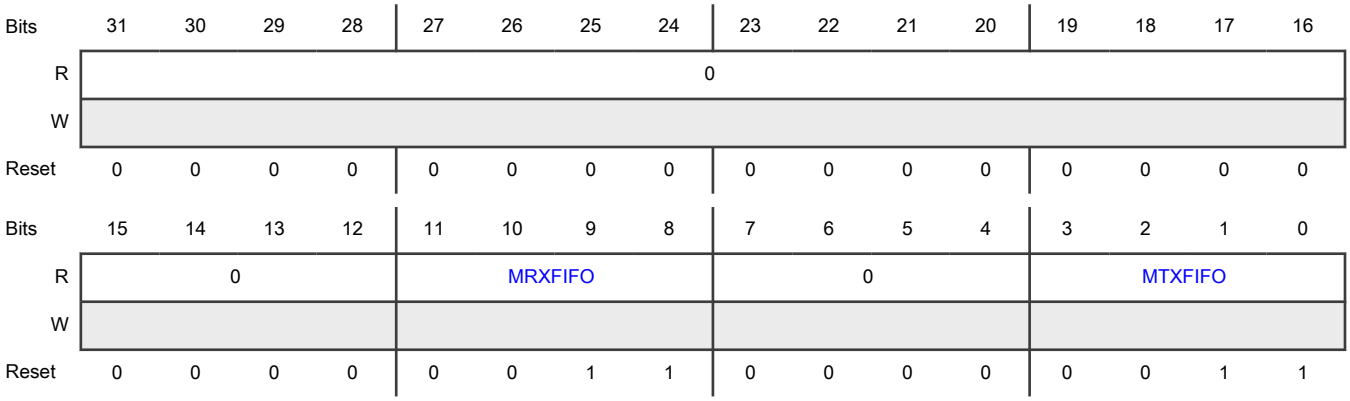
66.3.6.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function
Contains parameter values implemented in the module.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-12 —	Reserved
11-8 MRXFIFO	Controller Receive FIFO Size Configures the number of words in the controller receive FIFO to $2^{MRXFIFO}$.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-4 —	Reserved
3-0 MTXFIFO	Controller Transmit FIFO Size Configures the number of words in the controller transmit FIFO to $2^{MTXFIFO}$.

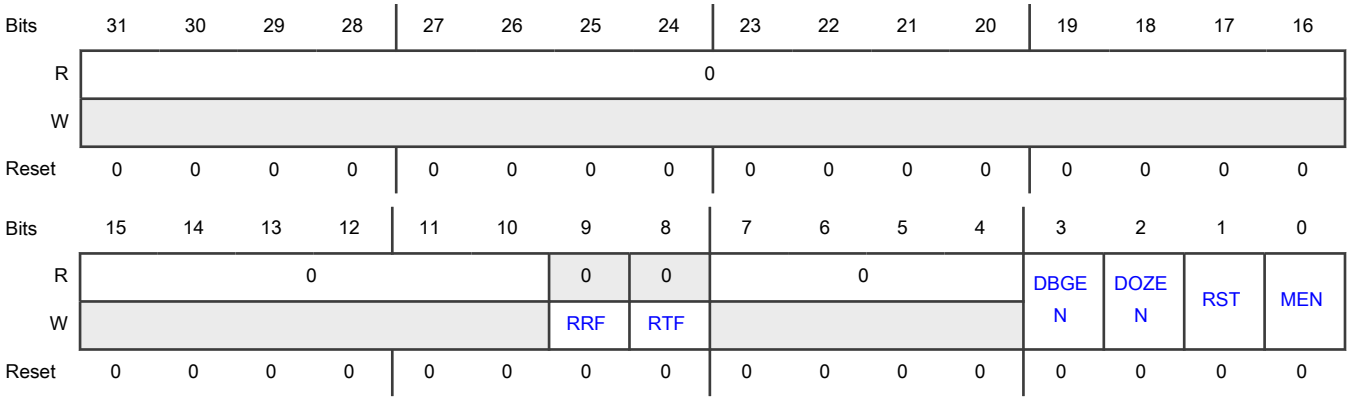
66.3.6.1.4 Controller Control (MCR)

Offset

Register	Offset
MCR	10h

Function
Contains resets, debug enable, and other controller control settings.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 RRF	Reset Receive FIFO Resets the receive FIFO. 0b - No effect 1b - Reset receive FIFO

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 RTF	Reset Transmit FIFO Resets the transmit FIFO. 0b - No effect 1b - Reset transmit FIFO
7-4 —	Reserved
3 DBGEN	Debug Enable Enables the controller in Debug mode. 0b - Disable 1b - Enable
2 DOZEN	Doze Mode Enable Enables the controller in Doze mode. 0b - Enable 1b - Disable
1 RST	Software Reset Resets all internal controller logic and registers except Controller Control (MCR). This field remains 1 (enabled) until you write 0 to it. The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset. 0b - No effect 1b - Reset
0 MEN	Controller Enable Enables the controller logic. 0b - Disable 1b - Enable

66.3.6.1.5 Controller Status (MSR)

Offset

Register	Offset
MSR	14h

Function
Contains status flags for transmit and receive data, for start and stop conditions, and for bus and controller busy or idle status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						BBF	MBF	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STF	DMF	PLTF	FEF	ALF	NDF	SDF	EPF	0						RDF	TDF
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Function
31-26 —	Reserved
25 BBF	Bus Busy Flag Specifies whether the I2C bus is busy. 0b - Idle 1b - Busy
24 MBF	Controller Busy Flag Specifies whether the I2C controller is busy. 0b - Idle 1b - Busy
23-16 —	Reserved
15 STF	Start Flag Specifies whether a Start condition is detected on the bus when the bus is idle. When MCFGR1[STARTCFG] is 0, this field becomes 1 only if the LPI2C controller is also idle. When MCFGR1[STARTCFG] is 1, STF becomes 1 for any Start condition when the I2C bus is idle. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Start condition not detected 1b - Start condition detected When writing

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Clear the flag
14 DMF	Data Match Flag Indicates whether the received data matches MDMR[MATCH0] or MDMR[MATCH1] (as configured by MCFGR1[MATCFG]). Received data discarded due to MTDR[CMD] does not cause this flag to set. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Matching data not received 1b - Matching data received When writing 0b - No effect 1b - Clear the flag
13 PLTF	Pin Low Timeout Flag Indicates whether pin low timeout has occurred. Sets when the SCL or SDA input is low for more than the number of PINLOW cycles defined by MCFGR3[PINLOW], even when the LPI2C controller is idle. You must resolve the pin low condition via software. PLTF cannot be cleared as long as the pin low timeout continues. Before LPI2C can initiate a Start condition, you must clear this flag. See MCFGR1[TIMECFG] for the SCL and/or SDA timeout settings. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Pin low timeout did not occur 1b - Pin low timeout occurred When writing 0b - No effect 1b - Clear the flag
12 FEF	FIFO Error Flag Detects the LPI2C controller's attempt to send or receive data without first generating a (repeated) Start condition. This error can occur when the transmit FIFO underflows when MCFGR1[AUTOSTOP] = 1. When this flag is set, the LPI2C controller sends a Stop condition (if busy). The controller does not initiate a new Start condition until the flag is cleared. **NOTE** This field behaves differently for register reads and writes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When reading 0b - No FIFO error 1b - FIFO error When writing 0b - No effect 1b - Clear the flag
11 ALF	Arbitration Lost Flag Indicates whether arbitration is lost. Either of these conditions sets this flag: • The LPI2C controller transmits a logic 1 and detects a logic 0 on the I2C bus. • The LPI2C controller detects a Start or Stop condition when the LPI2C controller is transmitting data. When ALF is set, the LPI2C controller releases the I2C bus (goes idle), and the LPI2C controller does not initiate a new Start condition until the ALF is cleared. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Controller did not lose arbitration 1b - Controller lost arbitration When writing 0b - No effect 1b - Clear the flag
10 NDF	NACK Detect Flag Indicates whether an unexpected NACK has been detected. This flag is set when the LPI2C controller detects a NACK it was not expecting when transmitting an address or data. When set, the controller does not initiate a new Start condition until this flag is cleared. If a NACK is expected for a given address (as configured by the command word), this flag is set if a NACK is not generated. When this flag is set, the LPI2C controller automatically transmits a Stop condition if MCFGR1[AUTOSTOP] = 1, or if the transmit FIFO is not empty. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No unexpected NACK detected 1b - Unexpected NACK detected When writing 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Clear the flag
9 SDF	Stop Detect Flag Indicates whether the LPI2C controller has generated a Stop condition. When MCFGR1[STOPCFG] = 0, this flag is set for any Stop condition. When MCFGR1[STOPCFG] = 1, this flag is set only when the LPI2C controller is also idle (transmit FIFO is empty). **NOTE** This field behaves differently for register reads and writes. When reading 0b - No Stop condition generated 1b - Stop condition generated When writing 0b - No effect 1b - Clear the flag
8 EPF	End Packet Flag Indicates whether the LPI2C controller has generated a repeated Start condition or a Stop condition. When the controller first generates a Start condition, this flag is not set. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No Stop or repeated Start generated 1b - Stop or repeated Start generated When writing 0b - No effect 1b - Clear the flag
7-2 —	Reserved
1 RDF	Receive Data Flag Indicates whether the receive data is ready. This flag is set when the number of words in the receive FIFO is greater than MFCR[RXWATER]. 0b - Receive data not ready 1b - Receive data ready
0 TDF	Transmit Data Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates whether transmit data is requested. This flag is set when the number of words in the transmit FIFO is equal or less than MFCR[TXWATER]. 0b - Transmit data not requested 1b - Transmit data requested

66.3.6.1.6 Controller Interrupt Enable (MIER)

Offset

Register	Offset
MIER	18h

Function

Contains enables for:

- Transmit and receive data interrupts
- Start, Stop, and NACK detection interrupts
- DMA interrupts

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STIE	DMIE	PLTIE	FEIE	ALIE	NDIE	SDIE	EPIE	0						RDIE	TDIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15 STIE	Start Interrupt Enable Enables interrupt for Start condition.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
14 DMIE	Data Match Interrupt Enable Enables interrupt for data match. 0b - Disable 1b - Enable
13 PLTIE	Pin Low Timeout Interrupt Enable Enables interrupt for pin-low timeout. 0b - Disable 1b - Enable
12 FEIE	FIFO Error Interrupt Enable Enables interrupt for FIFO error. 0b - Disable 1b - Enable
11 ALIE	Arbitration Lost Interrupt Enable Enables interrupt for arbitration lost. 0b - Disable 1b - Enable
10 NDIE	NACK Detect Interrupt Enable Enables interrupt for NACK detection. 0b - Disable 1b - Enable
9 SDIE	Stop Detect Interrupt Enable Enables interrupt for Stop detection. 0b - Disable 1b - Enable
8 EPIE	End Packet Interrupt Enable Enables interrupt for end packet. 0b - Disable 1b - Enable
7-2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1 RDIE	Receive Data Interrupt Enable Enables interrupt for receive data. 0b - Disable 1b - Enable
0 TDIE	Transmit Data Interrupt Enable Enables interrupt for transmit data. 0b - Disable 1b - Enable

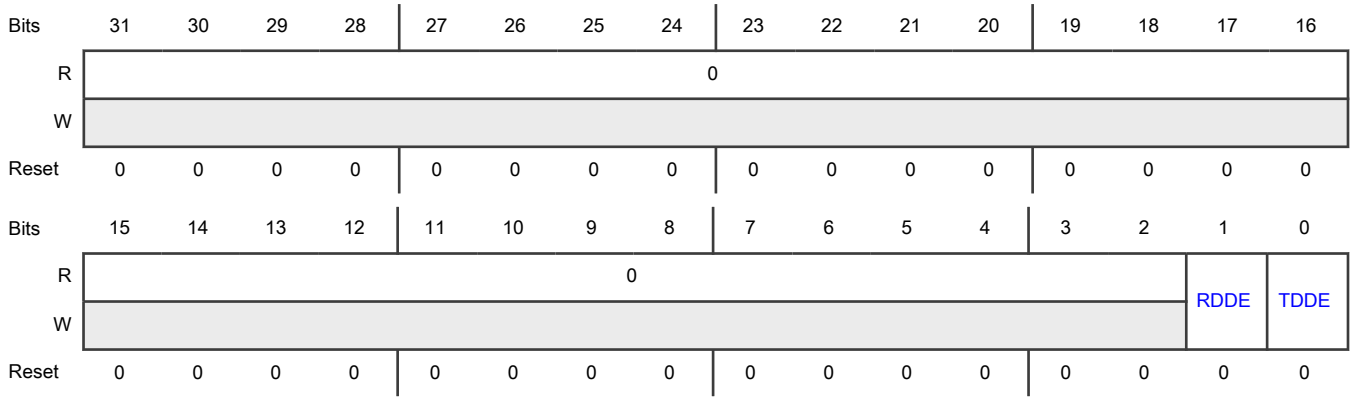
66.3.6.1.7 Controller DMA Enable (MDER)

Offset

Register	Offset
MDER	1Ch

Function
Contains DMA transmit, request, and receive enables.

Diagram



Fields

Field	Function
31-2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1 RDDE	Receive Data DMA Enable Enables DMA receive data. 0b - Disable 1b - Enable
0 TDDE	Transmit Data DMA Enable Enables DMA transmit data. 0b - Disable 1b - Enable

66.3.6.1.8 Controller Configuration 0 (MCFGR0)

Offset

Register	Offset
MCFGR0	20h

Function
Contains host settings and other receive and transfer settings.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0														ABORT	RELAX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						RDMO	CIRFIFO	0				HRDIR	HRSEL	HRPOL	HREN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-18	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
17 ABORT	Abort Transfer Determines whether the LPI2C controller completes the existing byte and then sends a Stop condition. A new transfer does not start while this field is 1. 0b - Normal transfer 1b - Abort existing transfer and do not start a new one
16 RELAX	Relaxed Mode Specifies the type of transfer. If this field is 1, the LPI2C controller relaxes several transfer restrictions. Transfers start when the bus is busy and FIFO commands are not checked for errors. You can use this mode to attempt recovery of a stuck I2C target by toggling the SCL pin. 0b - Normal transfer 1b - Relaxed transfer
15-10 —	Reserved
9 RDMO	Receive Data Match Only Determines whether all received data that does not set MSR[DMF] is discarded. After MSR[DMF] is set, the RDMO configuration is ignored. When disabling RDMO, write 0 to this field before writing 0 to MSR[DMF] to ensure that no receive data is lost. 0b - Received data is stored in the receive FIFO 1b - Received data is discarded unless MSR[DMF] is set
8 CIRFIFO	Circular FIFO Enable Enables the transmit FIFO read pointer to be saved to a temporary register. The transmit FIFO empties as normal. After the LPI2C controller is idle and the transmit FIFO is empty, the read pointer value is restored from the temporary register. This setting causes the contents of the transmit FIFO to be cycled through repeatedly. If MCFGR1[AUTOSTOP] is 1, then a Stop condition is sent whenever the transmit FIFO is empty and the read pointer is restored. 0b - Disable 1b - Enable
7-4 —	Reserved
3 HRDIR	Host Request Direction Configures the direction of the HREQ pin. When this field is 1, the LPI2C target drives the HREQ output pin and the pin becomes 1 when there is data in the target transmit data register. For proper operation when this field is 1, write 1 to SCFGR1[TXCFG].

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - HREQ pin is input (for LPI2C controller) 1b - HREQ pin is output (for LPI2C target)
2 HRSEL	Host Request Select Selects the source of the host request input. When host request is enabled, this field must not change. 0b - Host request input is pin HREQ 1b - Host request input is input trigger
1 HRPOL	Host Request Polarity Configures the polarity of the host request input. When host request is enabled, this field must not change. HRPOL sets the polarity for both the HREQ pin and the input trigger. • When HRPOL=0, the polarity is configured for active low, so host request is asserted if the HREQ pin or input trigger are logic 0. • When HRPOL=1, the polarity is configured for active high, so host request is asserted if the HREQ pin or input trigger are logic 1. 0b - Active low 1b - Active high
0 HREN	Host Request Enable Enables host request. When enabled, the LPI2C controller only initiates a Start condition if the host request input is asserted and the bus is idle. A repeated Start condition is not affected by the host request. 0b - Disable 1b - Enable

66.3.6.1.9 Controller Configuration 1 (MCFGR1)

Offset

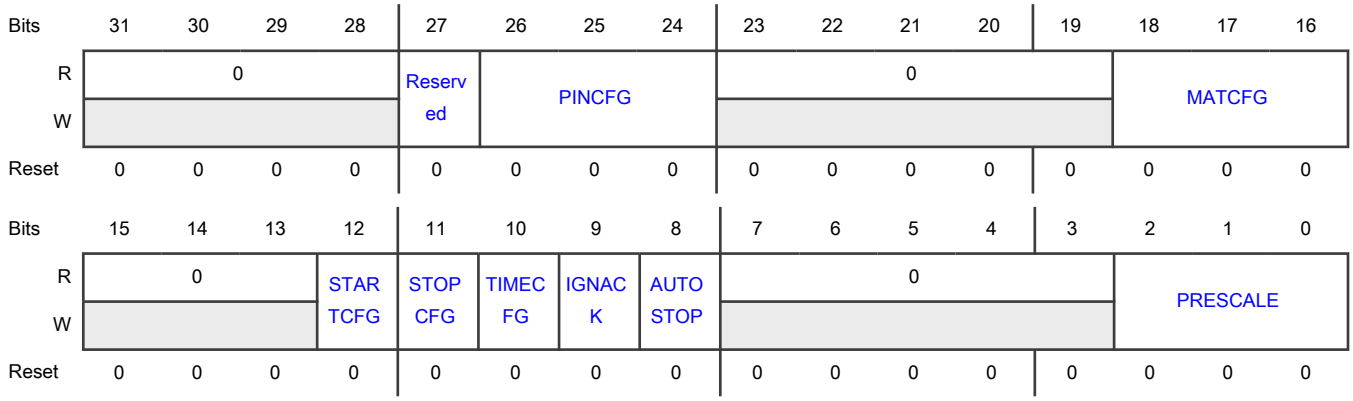
Register	Offset
MCFGR1	24h

Function

Contains controls for pin configuration, clock prescaler, and various other control settings.

Write to this register only when the I2C controller is disabled.

Diagram



Fields

Field	Function
31-28 —	Reserved
27 —	Reserved
26-24 PINCFG	Pin Configuration Configures the pin mode for LPI2C. 000b - Two-pin open drain mode. SCL/SDA pins: Bidirectional open drain for controller and target. SCLS/SDAS pins: Not used. 001b - Two-pin output only mode (Ultra-Fast mode). SCL/SDA pins: Output-only (Ultra-Fast mode) open drain for controller and target. SCLS/SDAS pins: Not used. 010b - Two-pin push-pull mode. SCL/SDA pins: Bidirectional push-pull for controller and target. SCLS/SDAS pins: Not used. 011b - Four-pin push-pull mode. SCL/SDA pins: Input only for controller and target. SCLS/SDAS pins: Output-only push-pull for controller and target. 100b - Two-pin open-drain mode with separate LPI2C target. SCL/SDA pins: Bidirectional open drain for controller. SCLS/SDAS pins: Bidirectional open drain for target. 101b - Two-pin output only mode (Ultra-Fast mode) with separate LPI2C target. SCL/SDA pins: Output-only (Ultra-Fast mode) open drain for controller. SCLS/SDAS pins: Output-only open drain for target. 110b - Two-pin push-pull mode with separate LPI2C target. SCL/SDA pins: Bidirectional push-pull for controller. SCLS/SDAS pins: Bidirectional push-pull for target. 111b - Four-pin push-pull mode (inverted outputs). SCL/SDA pins: Input only for controller and target. SCLS/SDAS pins: Inverted output-only push-pull for controller and target.
23-19 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-16 MATCFG	Match Configuration Configures the condition that sets MSR[DMF]. See Controller Data Match (MDMR). 000b - Match is disabled 001b - Reserved 010b - Match is enabled: first data word equals MDMR[MATCH0] OR MDMR[MATCH1] 011b - Match is enabled: any data word equals MDMR[MATCH0] OR MDMR[MATCH1] 100b - Match is enabled: (first data word equals MDMR[MATCH0]) AND (second data word equals MDMR[MATCH1) 101b - Match is enabled: (any data word equals MDMR[MATCH0]) AND (next data word equals MDMR[MATCH1) 110b - Match is enabled: (first data word AND MDMR[MATCH1]) equals (MDMR[MATCH0] AND MDMR[MATCH1]) 111b - Match is enabled: (any data word AND MDMR[MATCH1]) equals (MDMR[MATCH0] AND MDMR[MATCH1])
15-13 —	Reserved
12 STARTCFG	Start Configuration Configures the conditions that set MSR[STF] when a Start condition occurs. When this field is 0, MSR[STF] is set on a Start condition when both the I2C bus and LPI2C are idle. In other words, any nonrepeated Start condition is initiated by any controller on the bus except the LPI2C controller. When this field is 1, MSR[STF] is set on a Start condition when the I2C bus is idle. In other words, any nonrepeated Start condition is initiated by any controller on the bus including the LPI2C controller. 0b - Sets when both I2C bus and LPI2C controller are idle 1b - Sets when I2C bus is idle
11 STOPCFG	Stop Configuration Configures the conditions that set MSR[SDF]. When this field is 0, any Stop condition generated by the LPI2C controller sets SDF. When this field is 1, the last Stop condition before the LPI2C controller is idle sets SDF. In this case, the transmit FIFO is empty at the time of the Stop condition. 0b - Any Stop condition 1b - Last Stop condition
10 TIMECFG	Timeout Configuration Configures which signals must be low for longer than the configured timeout to set MSR[PLTF]. When this field is 0, MSR[PLTF] is set when SCL is low for longer than the configured timeout.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - SCL 1b - SCL or SDA
9 IGNACK	Ignore NACK Determines whether the LPI2C controller ignores a received NACK and treats it as an ACK. In this case, MSR[NDF] is never set. This field must be 1 in Ultra-Fast mode. 0b - No effect 1b - Treat a received NACK as an ACK
8 AUTOSTOP	Automatic Stop Generation Determines whether a Stop condition is generated when the LPI2C controller is busy and the transmit FIFO is empty. A Stop condition can also be generated using a transmit FIFO command. When this field is 1, a Stop condition is automatically generated when the transmit FIFO is empty and the LPI2C controller is busy. 0b - No effect 1b - Stop automatically generated
7-3 —	Reserved
2-0 PRESCALE	Prescaler Configures the clock prescaler used for all LPI2C controller logic except the digital glitch filters. 000b - Divide by 1 001b - Divide by 2 010b - Divide by 4 011b - Divide by 8 100b - Divide by 16 101b - Divide by 32 110b - Divide by 64 111b - Divide by 128

66.3.6.1.10 Controller Configuration 2 (MCFGR2)

Offset

Register	Offset
MCFGR2	28h

Function

Contains the configuration for the bus idle timeout and glitch filters for SDA and SCL.

Write to this register only when the I2C controller is disabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				FILTSDA				0				FILTSCL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				BUSIDLE											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 FILTSDA	Glitch Filter SDA Configures the I2C controller digital glitch filters for the SDA input. The latency through the glitch filter is equal to the number of cycles defined by this field. The value of this field must be less than the minimum SCL low or high period. Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored. Writing 0 to this field disables the glitch filter. MCFGR1[PRESCALE] does not affect the glitch filter cycle count. It is automatically bypassed in HS mode.
23-20 —	Reserved
19-16 FILTSCL	Glitch Filter SCL Configures the I2C controller digital glitch filters for SCL input. The latency through the glitch filter is equal to the number of cycles defined by this field. The value of this field must be less than the minimum SCL low or high period. Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored. These cycles are based on the functional clock. Writing 0 to this field disables the glitch filter. MCFGR1[PRESCALE] does not affect the glitch filter cycle count. It is automatically bypassed in HS mode.
15-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-0 BUSIDLE	Bus Idle Timeout Configures the bus idle timeout period, in clock cycles. If both SCL and SDA are high for longer than the number of cycles defined by this field, the I2C bus is assumed to be idle and the controller can generate a Start condition. Writing 0 to this field disables the bus idle timeout.

66.3.6.1.11 Controller Configuration 3 (MCFGR3)

Offset

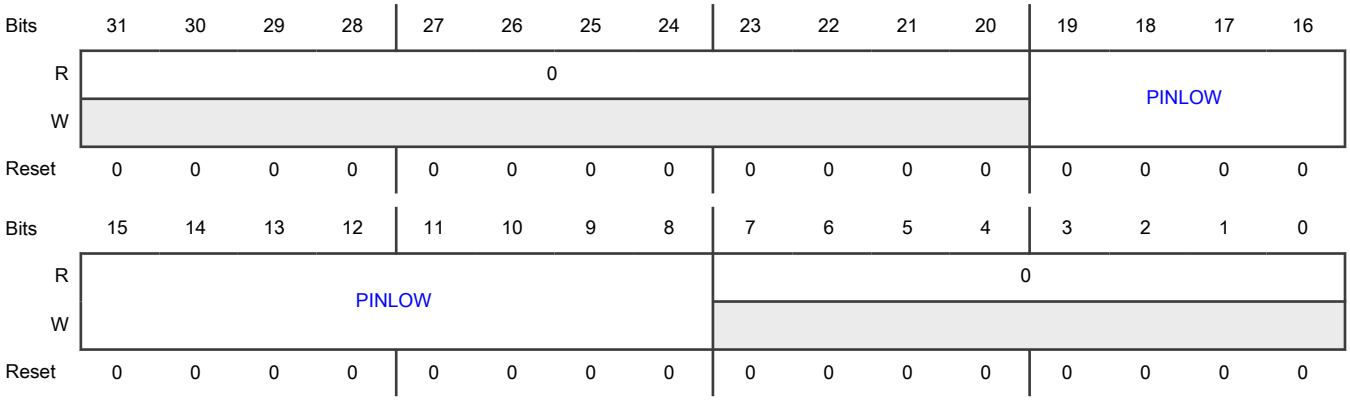
Register	Offset
MCFGR3	2Ch

Function

Configures the threshold value for the pin low timeout flag.

Write to this register only when the I2C controller is disabled.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-8 PINLOW	Pin Low Timeout Configures the threshold value, in clock cycles, that sets MSR[PLTF].

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If SCL or SDA (selected by MCFGR1[TIMECFG]) is low for longer than (PINLOW × 256) cycles, MSR[PLTF] is set. When this field is 0, the pin low timeout feature is disabled.
7-0 —	Reserved

66.3.6.1.12 Controller Data Match (MDMR)

Offset

Register	Offset
MDMR	40h

Function

Contains data match values.

Write to this register only when the I2C controller is disabled or idle.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								MATCH1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								MATCH0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-16 MATCH1	Match 1 Value Specifies match 1 value that is compared to the received data when receive data match is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-8 —	Reserved
7-0 MATCH0	Match 0 Value Specifies match 0 value that is compared to the received data when receive data match is enabled.

66.3.6.1.13 Controller Clock Configuration 0 (MCCR0)

Offset

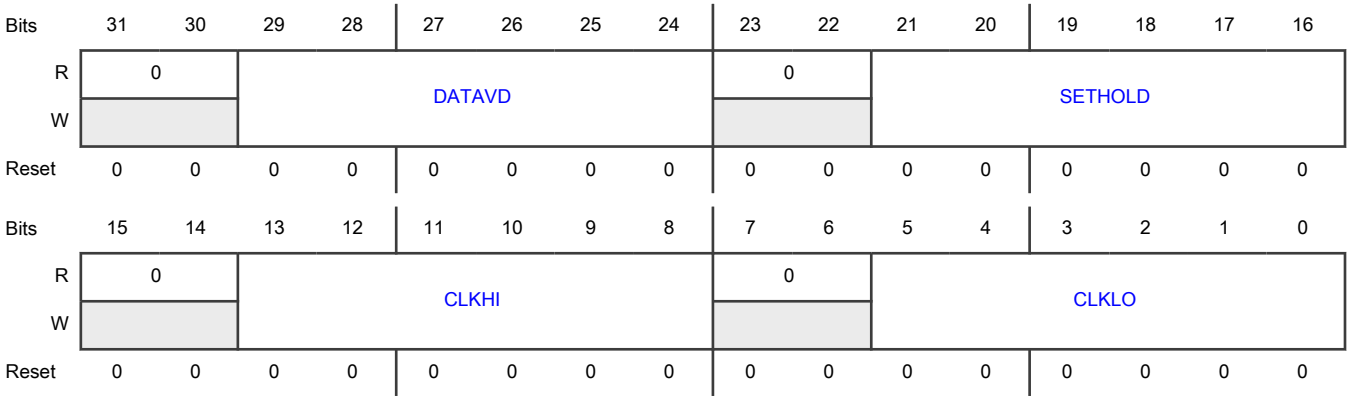
Register	Offset
MCCR0	48h

Function

Configures various clock controls.

You cannot make changes to this register when the I2C controller is enabled and is used for standard, fast, fast-mode plus, and ultra-fast transfers.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-24 DATAVD	Data Valid Delay

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Specifies the minimum number of cycles (minus one) used as the data hold time for SDA. This value must be less than the minimum SCL low period.
23-22 —	Reserved
21-16 SETHOLD	Setup Hold Delay Specifies the minimum number of cycles (minus one) used by the controller for these conditions: • Hold time for a Start • Setup and hold time for a repeated Start • Setup time for a Stop The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles.
15-14 —	Reserved
13-8 CLKHI	Clock High Period Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock high. The SCL high time is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles.
7-6 —	Reserved
5-0 CLKLO	Clock Low Period Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock low. This value is also used for the minimum bus free time between a Stop and a Start condition. This period is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles.

66.3.6.1.14 Controller Clock Configuration 1 (MCCR1)

Offset

Register	Offset
MCCR1	50h

Function

Configures various clock controls.

You cannot makes changes to this register when the I2C controller is enabled and is used for HS mode transfers. The separate clock configuration for HS mode allows arbitration to take place in Fast mode (with timing configured by Controller Clock Configuration 0 (MCCR0)), before switching to HS mode (with timing configured by MCCR1).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		DATAVD						0		SETHOLD					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		CLKHI						0		CLKLO					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29-24 DATAVD	Data Valid Delay Specifies the minimum number of cycles (minus one) used as the data hold time for SDA. This value must be less than the minimum SCL low period.
23-22 —	Reserved
21-16 SETHOLD	Setup Hold Delay Specifies the minimum number of cycles (minus one) used by the controller for these conditions: • Hold time for a Start condition • Setup and hold time for a repeated Start condition • Setup time for a Stop condition The setup time is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles.
15-14 —	Reserved
13-8 CLKHI	Clock High Period Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock high. The SCL high time is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring

Table continues on the next page...

Table continued from the previous page...

Field	Function
	any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles.
7-6 —	Reserved
5-0 CLKLO	Clock Low Period Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock low. This value is also used for the minimum bus free time between a Stop and a Start condition. This period is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles.

66.3.6.1.15 Controller FIFO Control (MFCR)

Offset

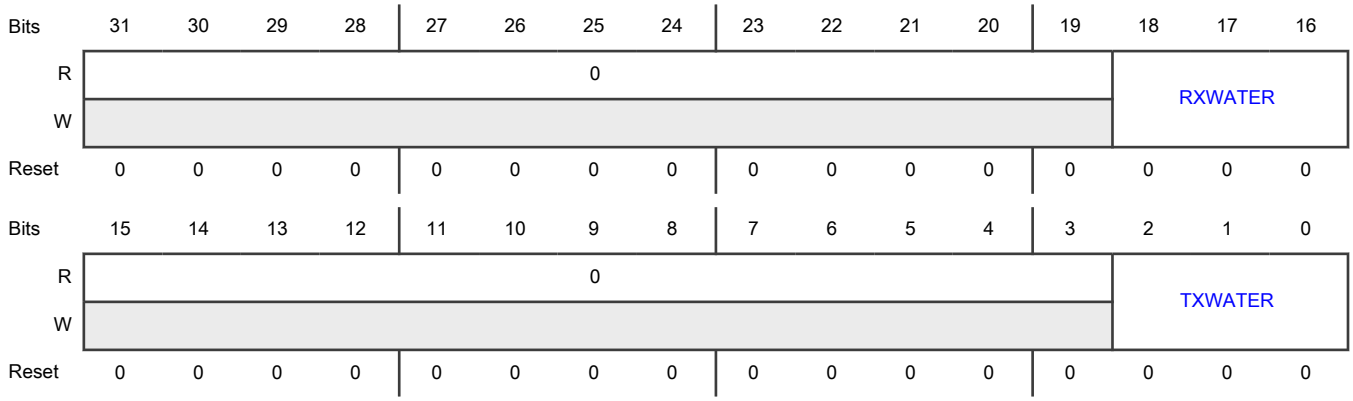
Register	Offset
MFCR	58h

Function

Controls the receive and transmit FIFO watermark values.

This register is used only in Stop mode, when this register is static (not changing).

Diagram



Fields

Field	Function
31-19	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
18-16 RXWATER	Receive FIFO Watermark Determines the watermark for setting SSR[RDF]. That flag is set when the number of words in the receive FIFO is greater than the value of this field. Writing a value equal to or greater than the FIFO size truncates the value.
15-3 —	Reserved
2-0 TXWATER	Transmit FIFO Watermark Determines the watermark for setting SSR[TDF]. That flag is set when the number of words in the transmit FIFO is equal or less than the value of this field. Writing a value equal to or greater than the FIFO size truncates the value.

66.3.6.1.16 Controller FIFO Status (MFSR)

Offset

Register	Offset
MFSR	5Ch

Function
Specifies the number of words in the transmit and receive FIFOs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												RXCOUNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												TXCOUNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-16 RXCOUNT	Receive FIFO Count Specifies the number of words in the receive FIFO.
15-4 —	Reserved
3-0 TXCOUNT	Transmit FIFO Count Specifies the number of words in the transmit FIFO.

66.3.6.1.17 Controller Transmit Data (MTDR)

Offset

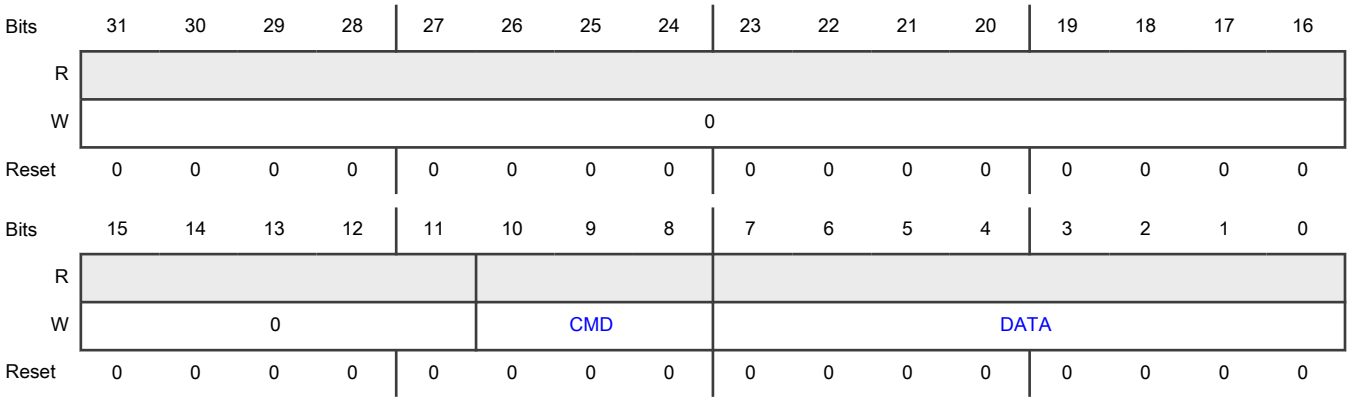
Register	Offset
MTDR	60h

Function

Configures transmit data:

- An 8-bit write to MTDR[CMD] is ignored and does not increment the FIFO write pointer.
- An 8-bit write to MTDR[DATA] zero-extends the value of MTDR[CMD] and increments the FIFO write pointer.
- A 16-bit or 32-bit write operation writes to both MTDR[CMD] and MTDR[DATA] and increments the FIFO write pointer.

Diagram



Fields

Field	Function
31-11 —	Reserved
10-8 CMD	Command Data Selects command transmitted by controller. 000b - Transmit the value in DATA[7:0] 001b - Receive (DATA[7:0] + 1) bytes. DATA[7:0] is used as a byte counter. Receive that many bytes and check each for a data match (if configured) before storing the received data in the receive FIFO. 010b - Generate Stop condition on I2C bus 011b - Receive and discard (DATA[7:0] + 1) bytes. DATA[7:0] is used as a byte counter. Receive that many bytes but do not check for a data match or store those bytes in the receive FIFO. 100b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0] 101b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0] (this transfer expects a NACK to be returned) 110b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0] using HS mode 111b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0] using HS mode (this transfer expects a NACK to be returned)
7-0 DATA	Transmit Data Contains data used by the commands listed in MTDR[CMD]. Performing an 8-bit write to this field zero-extends the value of MTDR[CMD].

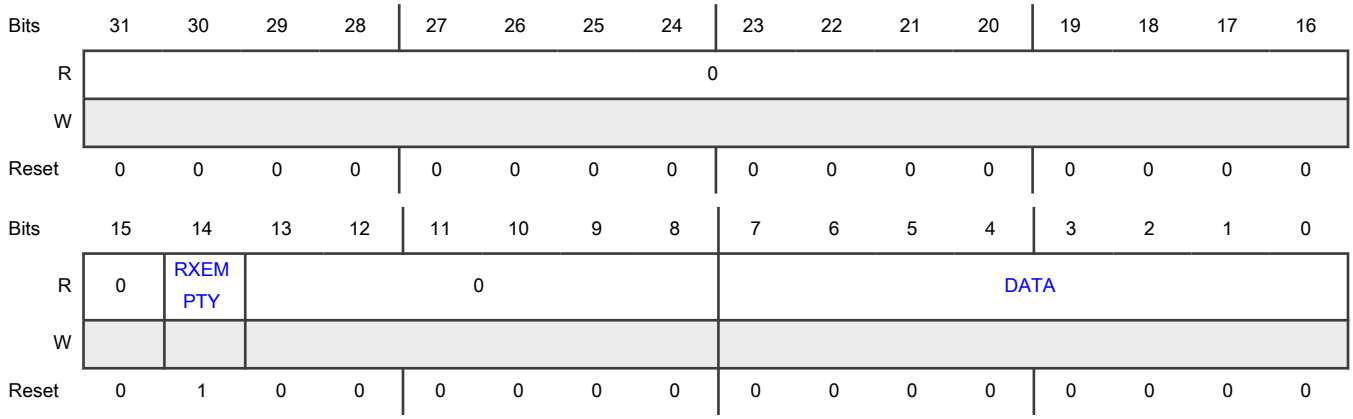
66.3.6.1.18 Controller Receive Data (MRDR)

Offset

Register	Offset
MRDR	70h

Function
Contains the status of the receive FIFO and the data received by the I2C controller that has not been discarded.

Diagram



Fields

Field	Function
31-15 —	Reserved
14 RXEMPTY	Receive Empty Indicates whether the controller receive data FIFO is empty. 0b - Not empty 1b - Empty
13-8 —	Reserved
7-0 DATA	Receive Data Contains data received by the I2C controller that has not been discarded. Received data can be discarded due to the command in MTDR[CMD], or the controller can be configured to discard nonmatching data.

66.3.6.1.19 Controller Receive Data Read Only (MRDROR)

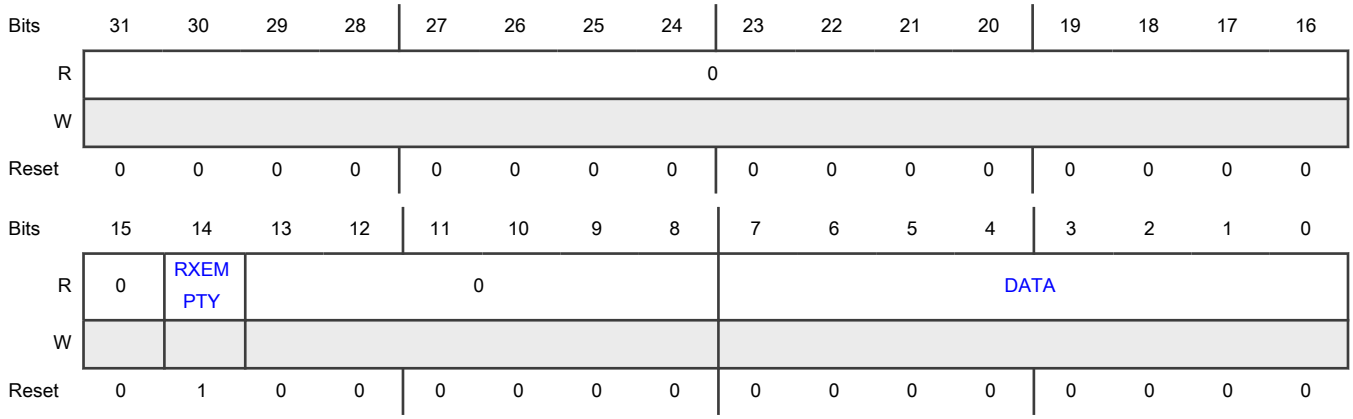
Offset

Register	Offset
MRDROR	78h

Function

Contains the status of the receive FIFO and returns the data received by the I2C controller, but does not pull the data from the FIFO.

Diagram



Fields

Field	Function
31-15 —	Reserved
14 RXEMPTY	RX Empty Indicates whether the receive data FIFO is empty. 0b - Not empty 1b - Empty
13-8 —	Reserved
7-0 DATA	Receive Data

66.3.6.1.20 Target Control (SCR)

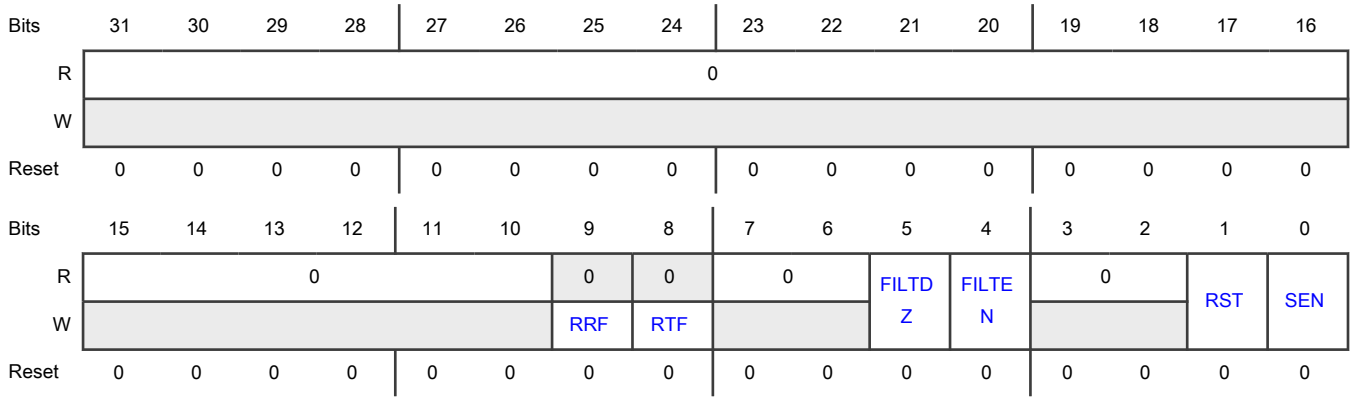
Offset

Register	Offset
SCR	110h

Function

Contains resets and other target control settings.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 RRF	Reset Receive FIFO Empties the receive FIFO in Target Receive Data (SRDR). 0b - No effect 1b - SRDR is now empty
8 RTF	Reset Transmit FIFO Empties the transmit FIFO in Target Transmit Data (STDR). 0b - No effect 1b - STDR is now empty
7-6 —	Reserved
5 FILTDZ	Filter Doze Enable Enables filter in Doze mode. Update this field only when the I2C target is disabled. 0b - Enable 1b - Disable
4 FILTEN	Filter Enable Enables digital filter and output delay counter for target mode. Update this field only when the I2C target is disabled. 0b - Disable 1b - Enable
3-2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1 RST	Software Reset Resets target mode logic. The reset takes effect immediately. The value of this field remains 1 until you write 0 to it. There is no minimum delay required before clearing the software reset. 0b - Not reset 1b - Reset
0 SEN	Target Enable Enables I2C Target mode. 0b - Disable 1b - Enable

66.3.6.1.21 Target Status (SSR)

Offset

Register	Offset
SSR	114h

Function

Contains status flags for transmit and receive data, for error conditions, and for bus and target busy or idle status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						BBF	SBF	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SARF	GCF	AM1F	AM0F	FEF	BEF	SDF	RSF	0				TAF	AVF	RDF	TDF
W					W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25 BBF	Bus Busy Flag Indicates whether an I2C bus is idle or busy. 0b - Idle 1b - Busy
24 SBF	Target Busy Flag Indicates whether an I2C target is idle or busy. 0b - Idle 1b - Busy
23-16 —	Reserved
15 SARF	SMBus Alert Response Flag Indicates whether an SMBus alert response has been detected. You can clear this flag by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup. 0b - Disabled or not detected 1b - Enabled and detected
14 GCF	General Call Flag Indicates whether a target has detected the general call address. You can clear this flag by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup. 0b - General call address disabled or not detected 1b - General call address detected
13 AM1F	Address Match 1 Flag Indicates whether the received address matches the value in ADDR1, or it falls within the ADDR0 to ADDR1 range as configured by SCFGR1[ADDRCFG]. This flag is cleared by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup. 0b - Matching address not received 1b - Matching address received
12	Address Match 0 Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
AM0F	Indicates whether the received address matches the ADDR0 field, as configured by SCFGR1[ADDRCFG]. This flag is cleared by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup. 0b - ADDR0 matching address not received 1b - ADDR0 matching address received
11 FEF	FIFO Error Flag Indicates whether there is a FIFO error. This flag can only be set when clock stretching is disabled. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No FIFO error 1b - FIFO error When writing 0b - No effect 1b - Clear the flag
10 BEF	Bit Error Flag Indicates whether the LPI2C target has transmitted a logic 1 and detects a logic 0 on the I2C bus. The target ignores the rest of the transfer until the next (repeated) Start condition. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No bit error occurred 1b - Bit error occurred When writing 0b - No effect 1b - Clear the flag
9 SDF	Stop Detect Flag Indicates whether the LPI2C target detects a Stop condition, and if the LPI2C target matched the last address byte. When SCFGR1[SDCFG] = 1, this flag is set on any Stop condition. **NOTE** This field behaves differently for register reads and writes. When reading

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No Stop detected 1b - Stop detected When writing 0b - No effect 1b - Clear the flag
8 RSF	Repeated Start Flag Indicates whether the LPI2C target detects a repeated Start condition and if the LPI2C target matched the last address byte. When SCFGR1[RSCFG] = 1, this flag is set on any repeated Start condition. This flag is not set when the target first detects a Start condition. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No repeated Start detected 1b - Repeated Start detected When writing 0b - No effect 1b - Clear the flag
7-4 —	Reserved
3 TAF	Transmit ACK Flag Indicates whether a transmit ACK or NACK is required. You can clear this flag by writing to Target Transmit ACK (STAR). 0b - Not required 1b - Required
2 AVF	Address Valid Flag Indicates whether the contents of Target Address Status (SASR) are valid. You can clear this flag by reading SASR. When SCFGR1[RXCFG] = 1, this flag is also cleared by reading Target Receive Data (SRDR). 0b - Not valid 1b - Valid
1 RDF	Receive Data Flag Indicates whether receive data is ready. You can clear this flag by reading Target Receive Data (SRDR). When SCFGR1[RXCFG] = 1, this flag is not cleared when reading Target Receive Data (SRDR) if SSR[AVF] = 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not ready 1b - Ready
0 TDF	Transmit Data Flag Indicates whether transmit data has been requested. This flag is cleared by writing to Target Transmit Data (STDR). When SCFGR1[TXCFG] = 0, if a NACK, repeated Start, or Stop condition is detected, this flag is also cleared. 0b - Transmit data not requested 1b - Transmit data is requested

66.3.6.1.22 Target Interrupt Enable (SIER)

Offset

Register	Offset
SIER	118h

Function

Contains transmit and receive data interrupt enables, start and stop detect interrupt enables, and other target interrupt enables.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SARIE	GCIE	AM1IE	AM0IE	FEIE	BEIE	SDIE	RSIE	0				TAIE	AVIE	RDIE	TDIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 SARIE	SMBus Alert Response Interrupt Enable Enables interrupt for SMBus alert response. 0b - Disable 1b - Enable
14 GCIE	General Call Interrupt Enable Enables interrupt for general call. 0b - Disabled 1b - Enabled
13 AM1IE	Address Match 1 Interrupt Enable Enables interrupt for address match 1. 0b - Disable 1b - Enable
12 AM0IE	Address Match 0 Interrupt Enable Enables interrupt for address match 0. 0b - Disable 1b - Enable
11 FEIE	FIFO Error Interrupt Enable Enables interrupt for FIFO error. 0b - Disable 1b - Enable
10 BEIE	Bit Error Interrupt Enable Enables interrupt for bit error. 0b - Disable 1b - Enable
9 SDIE	Stop Detect Interrupt Enable Enables interrupt for Stop detection. 0b - Disable 1b - Enable
8 RSIE	Repeated Start Interrupt Enable Enables interrupt for repeated start. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
7-4 —	Reserved
3 TAIE	Transmit ACK Interrupt Enable Enables interrupt for transmit ACK. 0b - Disable 1b - Enable
2 AVIE	Address Valid Interrupt Enable Enables interrupt for valid address. 0b - Disable 1b - Enable
1 RDIE	Receive Data Interrupt Enable Enables interrupt for receive data. 0b - Disable 1b - Enable
0 TDIE	Transmit Data Interrupt Enable Enables interrupt for transmit data. 0b - Disable 1b - Enable

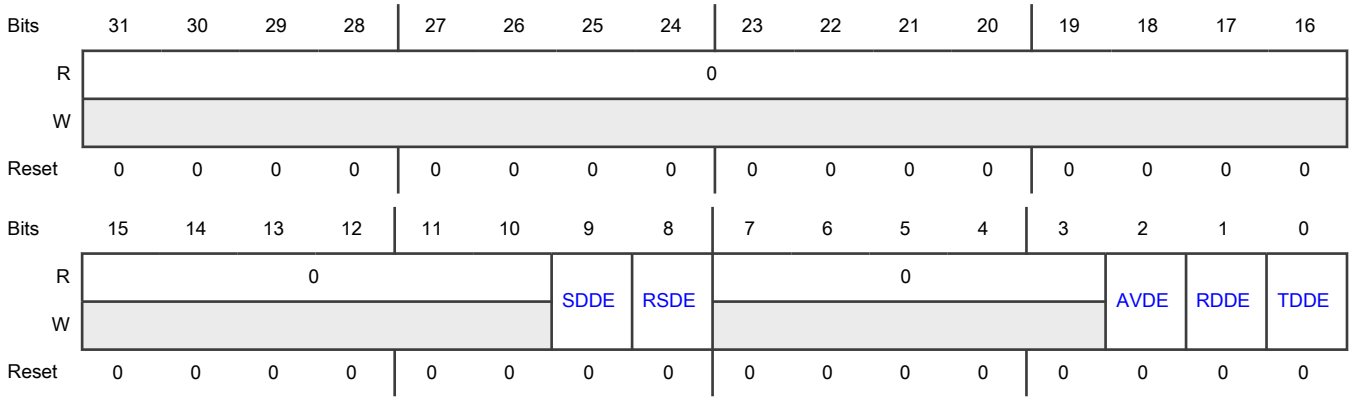
66.3.6.1.23 Target DMA Enable (SDER)

Offset

Register	Offset
SDER	11Ch

Function
Contains the transmit, request, and receive enables for DMA.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 SDDE	Stop Detect DMA Enable Enables DMA end-of-packet processing on stop detection. Reading Target Receive Data (SRDR) when it is empty: • Generates a DMA end-of-packet response. • Returns 0000_40FFh. • Writes 0 to MSR[SDF]. 0b - Disable 1b - Enable
8 RSDE	Repeated Start DMA Enable Enables DMA end-of-packet processing on repeated start. Reading Target Receive Data (SRDR) when it is empty: • Generates a DMA end-of-packet response. • Returns 0000_40FFh. • Writes 0 to MSR[RDF]. 0b - Disable 1b - Enable
7-3 —	Reserved
2 AVDE	Address Valid DMA Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Enables address valid DMA request. The address valid DMA request is shared with the receive data DMA request. If both are enabled, write 1 to SCFGR1[RXCFG] to allow the DMA to read the address from Target Receive Data (SRDR). 0b - Disable 1b - Enable
1 RDDE	Receive Data DMA Enable Enables receive data for DMA. 0b - Disable DMA request 1b - Enable DMA request
0 TDDE	Transmit Data DMA Enable Enables transmit data for DMA. 0b - Disable 1b - Enable

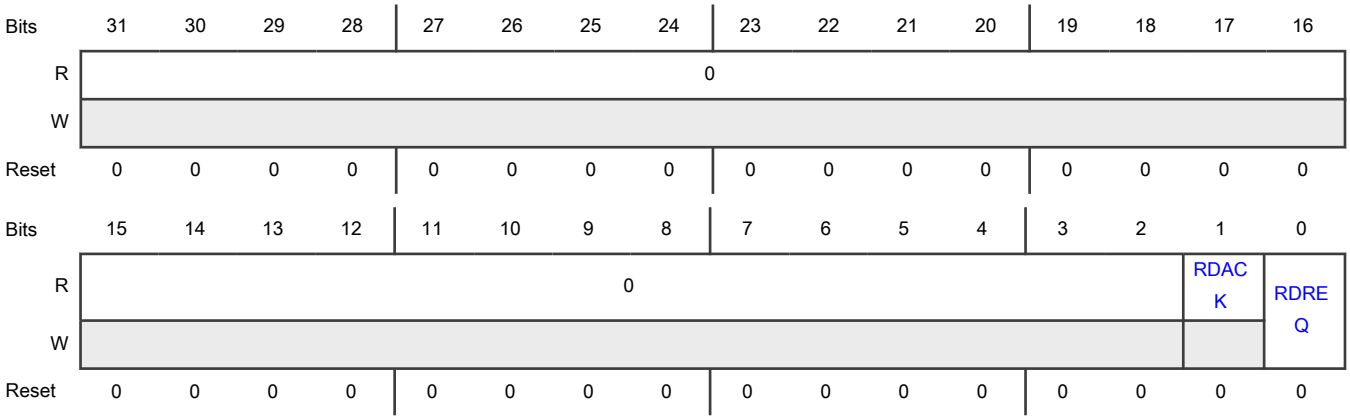
66.3.6.1.24 Target Configuration 0 (SCFGR0)

Offset

Register	Offset
SCFGR0	120h

Function
Configures the read request feature.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 RDACK	Read Acknowledge Flag Indicates whether a Start then Stop sequence with one SCL pulse between them acknowledged the read request while SCFGR0[RDREQ] = 1. You can clear this flag by writing 0 to SCFGR0[RDREQ]. 0b - Read Request not acknowledged 1b - Read Request acknowledged
0 RDREQ	Read Request Enables read request. When the I2C bus is idle, writing 1 to this field causes the LPI2C target to drive SDA low, triggering a Start condition. The LPI2C target releases SDA on the next falling edge of SCL or when you write 0 to this field. To initiate a second read request (for example, following I2C bus activity that did not acknowledge the request), write 0 then 1 to this field. When the I2C bus is busy, writing to this register always writes 0 to this field. 0b - Disable 1b - Enable

66.3.6.1.25 Target Configuration 1 (SCFGR1)

Offset

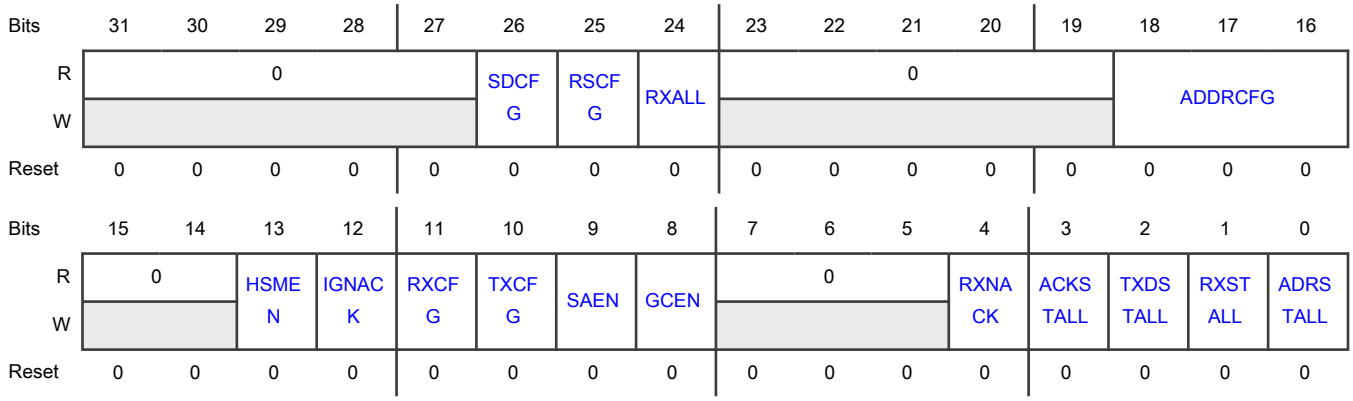
Register	Offset
SCFGR1	124h

Function

Configures various aspects of the target.

Write to this register only when the I2C target is disabled.

Diagram



Fields

Field	Function
31-27 —	Reserved
26 SDCFG	Stop Detect Configuration Configures the conditions that set MSR[SDF]. 0b - Any Stop condition following an address match 1b - Any Stop condition
25 RSCFG	Repeated Start Configuration Configures the conditions that set MSR[STF]. 0b - Any repeated Start condition following an address match 1b - Any repeated Start condition
24 RXALL	Receive All Enables receive-all functionality. When enabled, the LPI2C target stores all addresses on the I2C bus in Target Address Status (SASR) and all data on the I2C bus in Target Receive Data (SRDR). However, the LPI2C target does not drive SDA (for ACK or target-transmit transfer) unless there is an address match. The LPI2C target can drive SCL if clock stretching is enabled. When this field is 1, the LPI2C target only supports 7-bit addressing modes (you must configure SCFGR1[ADDRCFG] for 7-bit address match). Software can support 10-bit addresses, however. The first byte of a 10-bit address is saved to Target Address Match (SAMR). The second byte is saved as the first data byte in Target Receive Data (SRDR). Use this field to aid debugging of the I2C bus by saving all data on the bus without altering the state of the bus. When this field is 1, it is recommended to also write 1 to SCFGR1[RSCFG] and SCFGR1[SDCFG] so you can track the full state of the I2C bus. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-19 —	Reserved
18-16 ADDRCFG	Address Configuration Configures the condition that causes an address to match. 000b - Address match 0 (7-bit) 001b - Address match 0 (10-bit) 010b - Address match 0 (7-bit) or address match 1 (7-bit) 011b - Address match 0 (10-bit) or address match 1 (10-bit) 100b - Address match 0 (7-bit) or address match 1 (10-bit) 101b - Address match 0 (10-bit) or address match 1 (7-bit) 110b - From address match 0 (7-bit) to address match 1 (7-bit) 111b - From address match 0 (10-bit) to address match 1 (10-bit)
15-14 —	Reserved
13 HSMEN	HS Mode Enable Enables detection of the HS mode controller code of target address 0000_1XX, but does not cause an address match on this code. When this field is 1 and any HS mode controller code is detected, SCR[FILTEN] and SCFGR1[ACKSTALL] are ignored until the next Stop condition is detected. 0b - Disable 1b - Enable
12 IGNACK	Ignore NACK Determines whether the target ends transfer when a NACK condition is detected. When this field is 1, the LPI2C target continues transfers after a NACK is detected. This field is required to be 1 in Ultra-Fast mode. 0b - End transfer on NACK 1b - Do not end transfer on NACK
11 RXCFG	Receive Data Configuration Configures which data is returned and which flags are cleared when reading Target Receive Data (SRDR). When this field is 0, reading SRDR returns received data and clears MSR[RDF]. When this field is 1, reading SRDR: • Returns the value of Target Address Status (SASR) and clears SSR[AVF] when SSR[AVF] is set. • Returns received data and clears MSR[RDF] when SSR[AVF] is not set. 0b - Return received data, clear MSR[RDF]

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Return SASR and clear SSR[AVF] when SSR[AVF] is set, return received data and clear MSR[RDF] when SSR[AFV] is not set
10 TXCFG	Transmit Flag Configuration Determines which conditions set MSR[TDF]. This field always becomes 1 before a NACK is detected at the end of a target-transmit transfer. This change can cause an extra word to be written to the transmit data FIFO. When this field is 0, Target Transmit Data (STDR) is automatically emptied when a target-transmit transfer is detected. MSR[TDF] is set when a target-transmit transfer is detected, and MSR[TDF] is cleared at the end of the target-transmit transfer. When this field is 1, MSR[TDF] is set when STDR is empty, and MSR[TDF] is cleared when STDR is full. This setting allows STDR to be filled before a target-transmit transfer is detected. However, it can cause STDR to be written before a NACK is detected on the last byte of a target-transmit transfer. 0b - MSR[TDF] is set only during a target-transmit transfer when STDR is empty 1b - MSR[TDF] is set whenever STDR is empty
9 SAEN	SMBus Alert Enable Enables a match on an SMBus alert. 0b - Disable 1b - Enable
8 GCEN	General Call Enable Enables a general call address. 0b - Disable 1b - Enable
7-5 —	Reserved
4 RXNACK	Receive NACK Determines whether to override the setting of STAR[TXNACK] when the LPI2C receives a matching address during an overrun. When this field is 1, the LPI2C target responds with a NACK under the following conditions: • SSR[AVF] would be set due to matching an address, but that flag is already 1 (address overrun). • SSR[RDF] would be set due to receiving data, but that flag is already 1 (receive data overrun). 0b - ACK or NACK always determined by STAR[TXNACK] 1b - NACK always generated on address overrun or receive data overrun, otherwise ACK or NACK is determined by STAR[TXNACK]
3	ACK SCL Stall

Table continues on the next page...

Table continued from the previous page...

Field	Function
ACKSTALL	Enables SCL clock stretching during target-transmit address bytes and target-receiver address and data bytes, so you can write to Target Transmit ACK (STAR) before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the ninth bit, and is therefore not compatible with HS mode. If this field is 1: • You do not need to write 1 to SCFGR1[RXSTALL] or SCFGR1[ADRSTALL]. • When there is an address match on the first byte of a 10-bit address, SSR[AVF] is set, allowing you to read the received address before writing to Target Transmit ACK (STAR). 0b - Disable 1b - Enable
2 TXDSTALL	Transmit Data SCL Stall Enables SCL clock stretching when SSR[TDF] = 1 during a target-transmit transfer. Clock stretching occurs following the ninth bit, and is therefore compatible with HS mode. 0b - Disable 1b - Enable
1 RXSTALL	RX SCL Stall Enables SCL clock stretching when SSR[RDF] = 1 during a target-receive transfer. Clock stretching occurs following the ninth bit, and is therefore compatible with HS mode. 0b - Disable 1b - Enable
0 ADRSTALL	Address SCL Stall Enables SCL clock stretching when SSR[AVF] = 1. Clock stretching only occurs following the ninth bit, and is therefore compatible with HS mode. 0b - Disable 1b - Enable

66.3.6.1.26 Target Configuration 2 (SCFGR2)

Offset

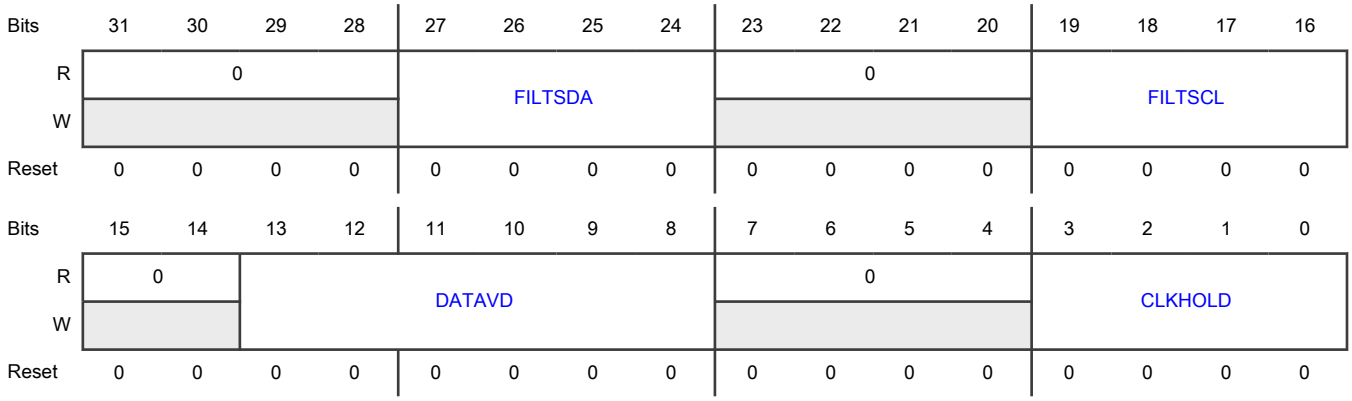
Register	Offset
SCFGR2	128h

Function

Configures data valid delay, clock hold time, and glitch filters for SDA and SCL.

Write to this register only when the I2C target is disabled.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 FILTSDA	Glitch Filter SDA Configures the I2C target digital glitch filters for SDA input. Writing 0 to this field disables the glitch filter. Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored. The latency through the glitch filter is equal to the number of cycles defined by this field + 3. The latency must be configured to be less than the minimum SCL low or high period. MCFGR1[PRESCALE] does not affect the glitch filter cycle count, and the glitch filter cycle count is disabled in HS mode.
23-20 —	Reserved
19-16 FILTSCL	Glitch Filter SCL Configures the I2C target digital glitch filters for SCL input. Writing 0 to this field disables the glitch filter. Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored. The latency through the glitch filter is equal to the number of cycles defined by this field + 3. The latency must be configured to be less than the minimum SCL low or high period. MCFGR1[PRESCALE] does not affect the glitch filter cycle count, and the glitch filter cycle count is disabled in HS mode.
15-14 —	Reserved
13-8	Data Valid Delay

Table continues on the next page...

Table continued from the previous page...

Field	Function
DATAVD	Configures the SDA data valid delay time for the I2C target, which is equal to FILTSCL + DATAVD + 3 cycles. The data valid delay must be configured to be less than the minimum SCL low period. MCFGR1[PRESCALE] does not affect the I2C target data valid delay time, and the I2C target data valid delay time is disabled in HS mode.
7-4 —	Reserved
3-0 CLKHOLD	Clock Hold Time Configures the minimum clock hold time for the I2C target, when clock stretching is enabled. The minimum hold time is equal to the number of cycles defined by this field + 3. MCFGR1[PRESCALE] does not affect the I2C target clock hold time, and the I2C target clock hold time is disabled in HS mode.

66.3.6.1.27 Target Address Match (SAMR)

Offset

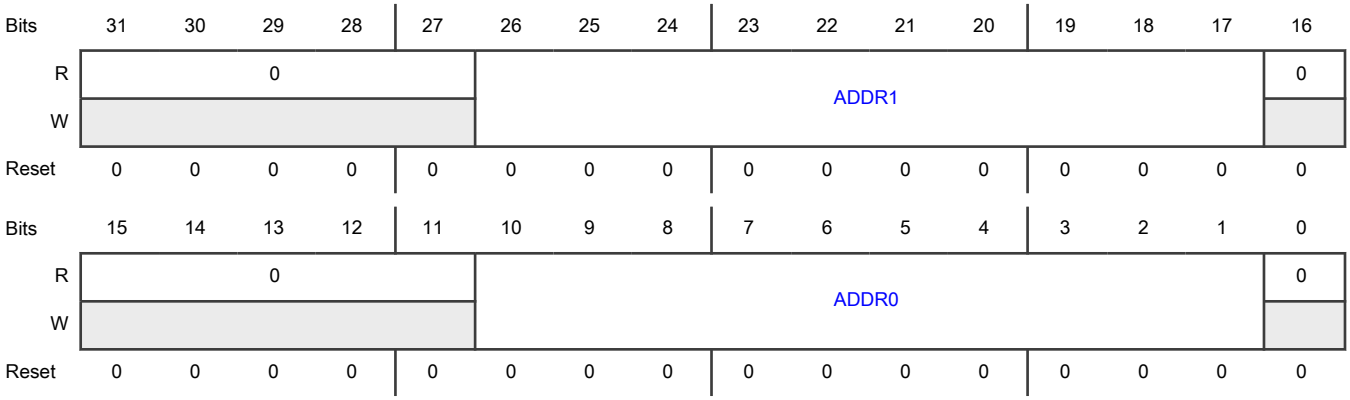
Register	Offset
SAMR	140h

Function

Contains address values for received target match comparison.

Write to this register only when the I2C target is disabled.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-17 ADDR1	Address 1 Value Contains the value of address 1, which is compared to the received address to detect the target address. In 10-bit mode, the first address byte is compared to {11110, ADDR1[26:25]} and the second address byte is compared to ADDR1[24:17]. In 7-bit mode, the address is compared to ADDR1[23:17].
16-11 —	Reserved
10-1 ADDR0	Address 0 Value Contains the value of address 0, which is compared to the received address to detect the target address. In 10-bit mode, the first address byte is compared to {11110, ADDR0[10:9]} and the second address byte is compared to ADDR0[8:1]. In 7-bit mode, the address is compared to ADDR0[7:1].
0 —	Reserved

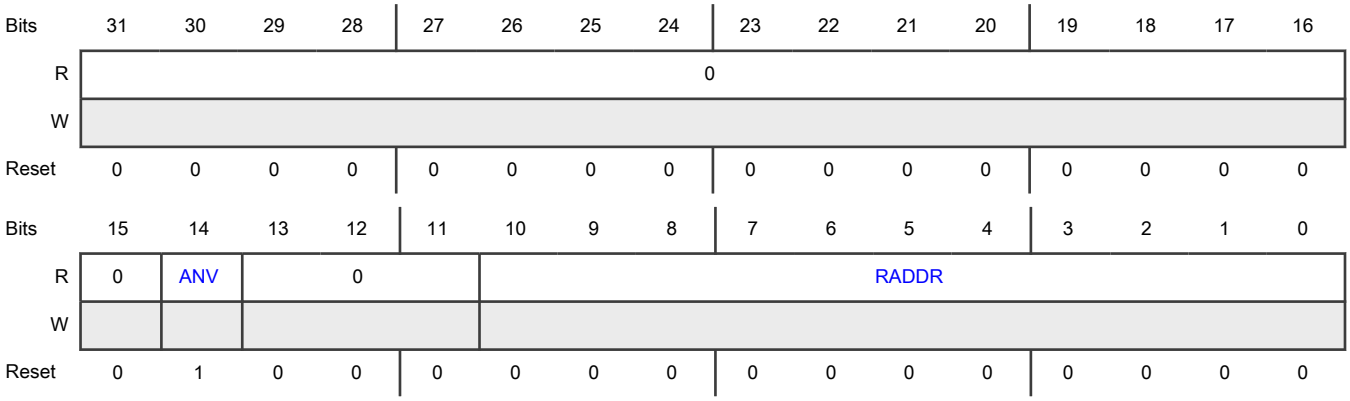
66.3.6.1.28 Target Address Status (SASR)

Offset

Register	Offset
SASR	150h

Function
Contains the received address and its validity.

Diagram



Fields

Field	Function
31-15 —	Reserved
14 ANV	Address Not Valid Indicates whether SASR[RADDR] is valid. 0b - Valid 1b - Not valid
13-11 —	Reserved
10-0 RADDR	Received Address Contains the received address. Updates whenever SSR[AM0F] or SSR[AM1F] is set. Reading Target Address Status (SASR) clears SSR[AM0F] and SSR[AM1F]. In 7-bit mode, the address byte is stored in RADDR[7:0]. In 10-bit mode, the first address byte is {11110, RADDR[10:9], RADDR[0]} and the second address byte is RADDR[8:1]. The Read-or-Write bit is therefore always stored in RADDR[0]. When SCFGR1[ACKSTALL] = 1, if the first address byte matches in 10-bit mode, the first address byte is stored in RADDR[7:0] so you can read this field before writing the Transmit ACK. If the second address byte matches, this field is then updated with the full 10-bit address.

66.3.6.1.29 Target Transmit ACK (STAR)

Offset

Register	Offset
STAR	154h

Function

Configures choice of ACK or NACK on each received word.

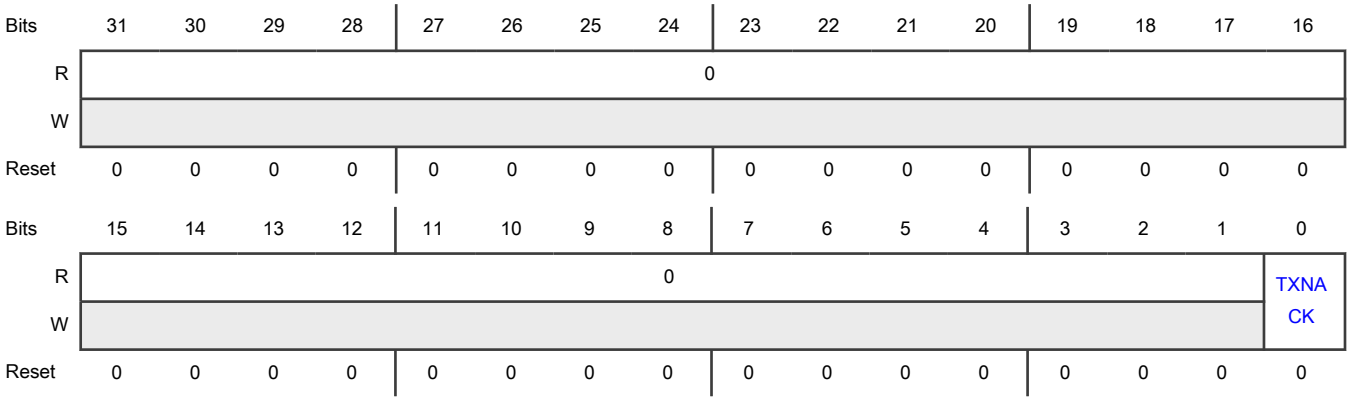
You can write to this register only when SCFGR1[ACKSTALL] = 1.

SCFGR1[ACKSTALL] enables clock stretching during the ACK-or-NACK bit slot. During this time, you can write to this register.

The logic ensures that the clock stretching continues for at least one bus clock cycle after this register is updated.

This clock stretching time can be extended via SCFGR2[CLKHOLD].

Diagram



Fields

Field	Function
31-1 —	Reserved
0 TXNACK	Transmit NACK Selects whether transmit ACK (logic 0) or NACK (logic 1) is returned on the bus by the I2C target after receiving each word. • When SCFGR1[ACKSTALL] = 1, a transmit NACK signal must be written once for each matching address byte and each received word. SCFGR1[ACKSTALL] must be 1, because that setting stalls the data transfer until software reads the received word (and determines whether to respond with an ACK or NACK). • To configure the default (ACK or NACK), you can write to this field when LPI2C target is disabled or idle. 0b - Transmit ACK 1b - Transmit NACK

66.3.6.1.30 Target Transmit Data (STDR)

Offset

Register	Offset
STDR	160h

Function

Contains the I2C target data to transmit.

Clock stretching (enabled or disabled) affects when the transmit data is transferred. SCFGR1[TXDSTALL] enables clock stretching during the first data bit of a target-transmit transfer.

If clock stretching is enabled (SCFGR1[TXDSTALL] = 1), the transmit data transfer is stalled until this register is updated. Clock stretching is extended by at least 1 bus clock cycle after this register is updated. Clock stretching can be delayed further by using SCFGR2[CLKHOLD].

If clock stretching is disabled (SCFGR1[TXDSTALL] = 0), the transmit data must be written before the start of the target-transmit transfer, otherwise SSR[FEF] is set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	0								DATA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 DATA	Transmit Data Contains the I2C target data to transmit. Writing data to this register stores I2C target transmit data in this register.

66.3.6.1.31 Target Receive Data (SRDR)

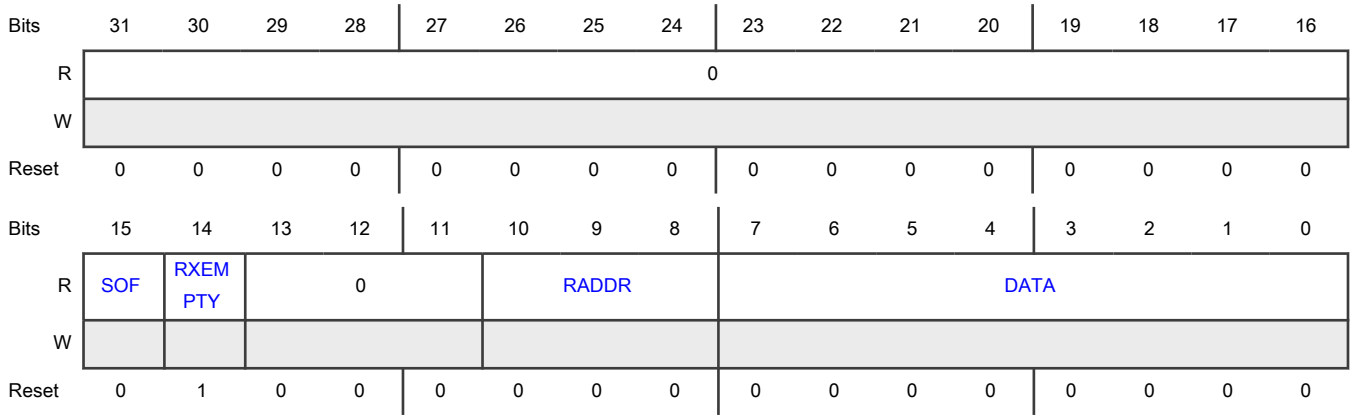
Offset

Register	Offset
SRDR	170h

Function

Contains status of target receive data transfer.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 SOF	Start of Frame Indicates whether this data word is the first data word since a (repeated) Start or Stop condition. 0b - Not first 1b - First
14 RXEMPTY	Receive Empty Indicates whether this register is empty. 0b - Not empty 1b - Empty
13-11 —	Reserved
10-8 RADDR	Received Address Contains the address received by the IC2 target. When both SCFGR1[RXCFG] and SSR[AVF] are 1, bits [10:8] of SASR[RADDR] are returned. Otherwise, this field returns zero.
7-0 DATA	Received Data Contains the data received by the I2C target. When both SCFGR1[RXCFG] and SSR[AVF] are 1, bits [7:0] of SASR[RADDR] are returned.

66.3.6.1.32 Target Receive Data Read Only (SRDROR)

Offset

Register	Offset
SRDROR	178h

Function

Contains the data received by the I2C target.

Reading this register returns the data received by the I2C target, but does not pull the data from the register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SOF	RXEMPTY	0			RADDR			DATA							
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15 SOF	Start of Frame Indicates whether this data word is the first data word since a (repeated) Start or Stop condition. 0b - Not the first 1b - First
14 RXEMPTY	Receive Empty Indicates whether Target Receive Data (SRDR) is empty. 0b - Not empty 1b - Empty
13-11 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
10-8 RADDR	Received Address Contains address received by the LPI2C target. When both SCFGR1[RXCFG] and SSR[AVF] are 1, bits [10:8] of SASR[RADDR] are returned. Otherwise, this field returns zero.
7-0 DATA	Receive Data Contains data received by the LPI2C target. When both SCFGR1[RXCFG] and SSR[AVF] are 1, bits [7:0] of SASR[RADDR] are returned.

66.3.6.1.33 Controller Transmit Command Burst (MTCBR0 - MTCBR127)

Offset

For n = 0 to 127:

Register	Offset
MTCBRn	200h + (n × 4h)

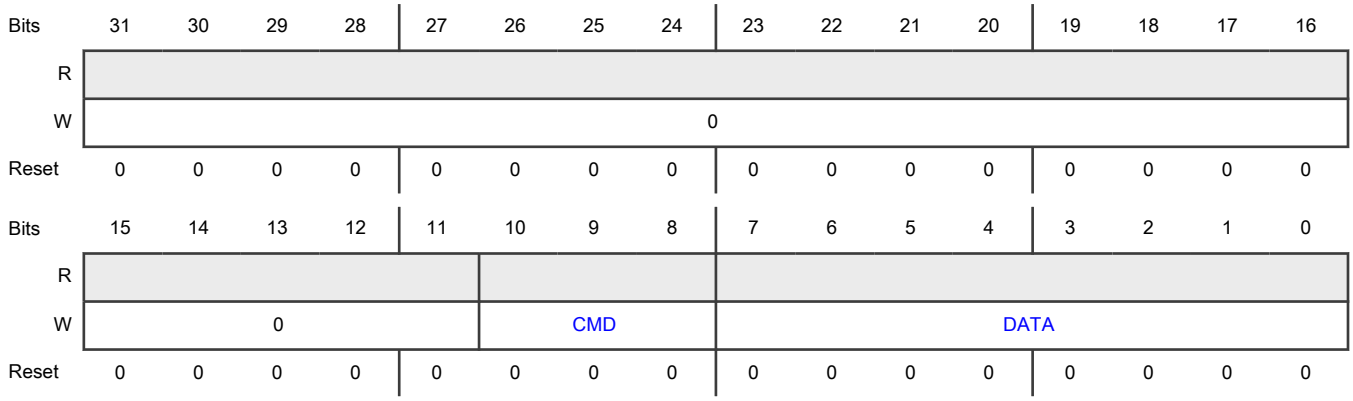
Function

Contains transmit Command and Data fields to support burst transfers.

Is an alias of the Controller Transmit Data Register designed to support incrementing burst transfers to the transmit FIFO by a DMA controller using aligned 8-bit, 16-bit, or 32-bit writes. The size of the Controller Transmit Data Burst Register is 512-bytes.

- An aligned 32-bit write in this region pushes one entry into the transmit FIFO.
- An aligned 16-bit write in this region to MTCBRn[15:0] pushes one entry into the transmit FIFO.
- An 8-bit write in this region to MTCBRn[7:0] updates DATA[7:0], but does not push the data into the transmit FIFO.
- An 8-bit write in this region to MTCBRn[15:8] pushes the data written to CMD[2:0] plus the previously written DATA[7:0] into the transmit FIFO.
- An 8-bit or 16-bit write in this region to MTXBRn[31:16] is ignored.

Diagram



Fields

Field	Function
31-11 —	Reserved
10-8 CMD	Command Command is written to the Controller Transmit Data Register.
7-0 DATA	Data Data is written to the Controller Transmit Data Register.

66.3.6.1.34 Transmit Data Burst (MTDBR0 - MTDBR252)

Offset

For n = 0 to 252:

Register	Offset
MTDBRn	400h + (n × 4h)

Function

Contains transmit Data fields to support burst transfers.

Is an alias of the Transmit Data Register designed to support incrementing burst transfers to the transmit FIFO by a DMA controller using 8-bit, 16-bit, or 32-bit writes. The CMD field is always zero-extended for transmit data. The size of the Transmit Data Burst Register is 1012 bytes.

- An aligned 32-bit write in this region pushes four zero-extended DATA entries into the transmit FIFO (DATA0 byte first). Note that the register access is extended by 3 wait states.
- An aligned 16-bit write in this region to either half of a 32-bit word pushes two zero-extended DATA entries into the transmit FIFO (DATA0 or DATA2 first). Note that the register access is extended by 1 wait state.
- An 8-bit write in this region pushes one zero-extended DATA entry into the transmit FIFO.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA3								DATA2							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 DATA3	Data Data is written to the MTDR[DATA] with MTDR[CMD] zero-extended.
23-16 DATA2	Data Data is written to the MTDR[DATA] with MTDR[CMD] zero-extended.
15-8 DATA1	Data Data is written to the MTDR[DATA] with MTDR[CMD] zero-extended.
7-0 DATA0	Data Data is written to the MTDR[DATA] with MTDR[CMD] zero-extended.

66.4 Low-Power Serial Peripheral Interface (LPSPI)

66.4.1 Overview

LPSPI provides an efficient interface (either as a master or slave) to an SPI bus, which is a synchronous serial communication interface used in embedded systems. It is typically used to perform short distance communications between microcontrollers and peripheral devices, on printed circuit boards. Typical applications include interfacing with secure digital cards and LCD displays.

66.4.1.1 Block diagram

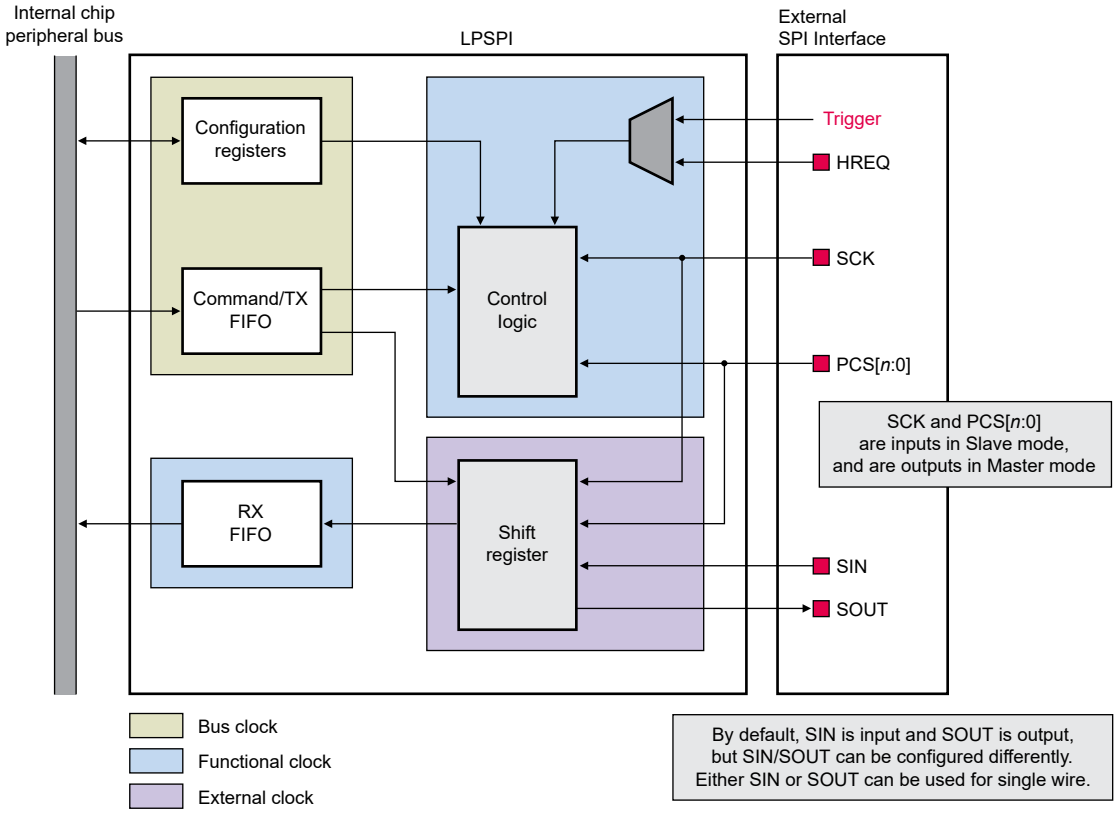


Figure 378. Block diagram

66.4.1.2 Features

- Minimal CPU overhead, with DMA transmit and receive requests supporting FIFO register accesses
- Operation continues in Deep Sleep mode, if configured to do so and an appropriate clock is available
- Support available for 32-bit word size
- Configurable clock polarity and phase
- Support available for 4 peripheral chip selects in Master mode
- Support available for Slave mode
- 8-word transmit and command FIFO
- 8-word receive FIFO
- Flexible timing parameters in Master mode, including SCK frequency and duty cycle, and delays between PCS and SCK edges
- Continuous transfer option to keep PCS asserted across multiple frames
- Full-duplex transfers that support 1-bit transmit and receive on each clock edge
- Half-duplex transfers that support:
  - 1-bit transmit or receive on each clock edge
  - 2-bit transmit or receive on each clock edge
  - 4-bit transmit or receive on each clock edge

- Option to use host request to control the start of an SPI bus transfer
- Receive data match logic that discards nonmatching data and interrupt on data match

66.4.2 Functional description

66.4.2.1 Master mode

66.4.2.1.1 Transmit and command FIFO commands

The transmit and command FIFO is a combined FIFO that includes both transmit data words and command words. You store:

- Transmit data words in the transmit and command FIFO, by writing to Transmit Data (TDR).
- Command words in the transmit and command FIFO, by writing to Transmit Command (TCR).

When a command word is at the top of the transmit and command FIFO, the actions that can occur depend on whether LPSPI is busy or between frames (see TCR[CONT] and TCR[CONTC]). See Table 562 for conditions and possible corresponding actions when a command word is at the top of the transmit and command FIFO.

Table 562. Possible actions when a command word is at the top of the transmit and command FIFO

Condition	Action
LPSPI is enabled and idle.	The command word is pulled from the FIFO, and this command word controls all subsequent transfers.
LPSPI is busy and TCR[CONTC] is 0.	The SPI frame completes at the end of the existing word, ignoring TCR[FRAMESZ]. The command word is then pulled from the FIFO and that command word controls all subsequent transfers (or until the next update to the command word). Note that a command word with TCR[CONTC] = 0 always terminates the existing transfer regardless of the previous TCR[CONT] value.
LPSPI is busy; the existing TCR[CONT] value is 1 and the new TCR[CONTC] value is 1.	The command word must be updated at the frame boundary. The command word is pulled from the FIFO during the last SCK pulse of the existing frame (based on the value of FRAMESZ), and the frame continues using the new command value for the rest of the frame (or until the next update to the command word). When TCR[CONTC] = 1, only the lower 24 bits of the command word are updated. If the command word is updated at a word boundary, then the transfer halts (stops) after that word. TCR[CONTC] is ignored when not at a frame boundary, so the frame ends prematurely.

TCR[CONT] = 1 keeps PCS asserted at end of frame, allowing the transfer to continue.

TCR[CONTC] = 1 specifies that this command word must not terminate the existing frame, and the transfer can continue using the new command word.

TCR[CONTC] = 1 is restricted in the sense that the new command must load on a frame boundary, and the only way for a transfer to continue from a frame boundary is when the previous command has TCR[CONT] = 1.

You can read the current state of the existing command word from Transmit Command (TCR). It requires at least three LPSPI functional clock cycles for Transmit Command (TCR) to update after you write to it (assuming an empty FIFO), and LPSPI must be enabled (CR[MEN] = 1).

Writing to Transmit Command (TCR) does not initiate an SPI bus transfer, unless TCR[TXMSK] = 1. When TCR[TXMSK] = 1, a new command word is not loaded until the end of the existing frame (based on the value of TCR[FRAMESZ]); at the end of the transfer, TCR[TXMSK] transitions to 0.

In Master mode, the LPSPI command word in Transmit Command (TCR) controls SPI attributes based on the selections in register fields. See Table 563 for TCR fields and associated functionality related to data transfer.

Table 563. Command word in Master mode

Transmit Command (TCR)		Description	Can this field be modified during a data transfer?
Field	Name		
CPOL	Clock polarity	Specifies the polarity of the SCK pin. Any change of CPOL value causes a transition on the SCK pin.	N
CPHA	Clock phase	Specifies the clock phase of the transfer.	N
PRESCALE	Prescaler value	Specifies a prescaler used to divide the LPSPI functional clock, to generate the timing parameters of the SPI bus transfer. Changing PRESCALE in conjunction with PCS enables LPSPI to connect to different slave devices at different frequencies.	N
PCS	Peripheral chip select	Specifies which PCS pin asserts for the transfer; the polarity of PCS is static and specified by CFGR1[PCSPOL]. If CFGR1[PCSCFG] = 1, do not select PCS[3:2].	N
LSBF	LSB first	Specifies whether LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first.	Y
BYSW	Byte swap	Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing with devices that organize data as big-endian.	Y
CONT	Continuous transfer	Configures LPSPI for a continuous transfer that keeps PCS asserted between frames (as specified by FRAMESZ). You must write a new command word to cause PCS to negate. Also, this field supports changing the command word at frame size boundaries.	Y
CONTC	Continuing command	Indicates that this is a new command word for the existing continuous transfer. When CONTC = 1, the command word must only be written to the transmit and command FIFO on a frame boundary.	Y
RXMSK	Receive data mask	Masks the receive data and does not store the masked receive data in the receive FIFO or perform receive data matching. This option is useful for half-duplex transfers or to specify which fields are compared during receive data matching.	Y
TXMSK	Transmit data mask	Masks the transmit data; masked transmit data is not pulled from the transmit FIFO, and the output data pin is 3-stated (unless otherwise configured by CFGR1[OUTCFG]). This option is useful for half-duplex transfers.	Y
WIDTH	Transfer width	Specifies the number of bits shifted on each SCK pulse: • 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats. • 2-bit and 4-bit half-duplex transfers are useful for interfacing with QuadSPI memory devices, and either TXMSK or RXMSK must also be 1.	Y
FRAMESZ	Frame size	Configures the frame size in number of bits equal to (FRAMESZ + 1):	Y

Table continues on the next page...

Table 563. Command word in Master mode (continued)

Transmit Command (TCR)		Description	Can this field be modified during a data transfer?
Field	Name		
		• The minimum frame size is 8 bits. • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32 bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately. • If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO contains the remaining bits. For example, a 72-bit transfer consists of three words: the first and second words are 32 bits, and the third word is 8 bits.	

66.4.2.1.1.1 SPI bus transfers

LPSPI initiates an SPI bus transfer when all these conditions are true:

- Data is written to the transmit FIFO.
- The HREQ pin is asserted (or the HREQ function is disabled).
- LPSPI is enabled.

To perform the SPI bus transfer, LPSPI uses the attributes configured in Transmit Command (TCR) and the timing parameters defined in Clock Configuration (CCR).

The SPI bus transfer ends after the number of bits indicated by the value of FRAMESZ have been transferred (provided CONT = 0), or at the end of a word when a new transmit command word is at the top of the transmit and command FIFO. When LPSPI is disabled, the SPI bus transfers end after the transmit FIFO is empty and LPSPI is idle.

The HREQ input is only checked when PCS is negated.

66.4.2.1.1.2 Circular FIFO

The transmit and command FIFO supports a circular FIFO feature. This feature enables the LPSPI master to (periodically) repeat a short data transfer that fits within the transmit and command FIFO, without requiring additional FIFO accesses. When the circular FIFO is enabled (CFGR0[CIRFIFO] = 1), the current state of the FIFO read pointer is saved and the status flags are not updated. After the FIFO is empty and LPSPI is idle, the FIFO read pointer is restored with the saved version, so the contents of the transmit and command FIFO are not permanently pulled from the FIFO when Circular FIFO mode is enabled.

66.4.2.1.2 Receive FIFO and data match

The receive FIFO stores received data during SPI bus transfers. When TCR[RXMSK] = 1, the received data is discarded instead of being stored in the receive FIFO:

- Received data is written to the receive FIFO when the last bit of the word is sampled.
- If the transmit FIFO is empty during a multiple-word or continuous transfer, then the receive data is written to the receive FIFO before the transfer stalls (assuming CFGR1[NOSTALL] = 0) while waiting for new transmit data or for a command word to be written.

LPSPI provides a receive data match function that can match received data against one of the two words in DMR0 and DMR1, or against a masked data word. You can also configure the received data match function to compare only the first one or two received data words since the start of the frame:

- Received data that is already discarded because of TCR[RXMSK] cannot cause the data match flag to set, and delays the receive data match on the first received data word, until all discarded data is received.

- You can configure the receive data match function to discard all received data until a data match is detected, using CFGR0[RDMO].
- After a receive data match, to allow all subsequent data to be received, write 0 to CFGR0[RDMO], and then write 0 to SR[DMF].

66.4.2.1.3 Timing parameters

The timing parameters that are used for all SPI bus transfers are relative to the LPSPI functional clock divided by the selection specified in TCR[PRESCALE]. Although you cannot change Clock Configuration (CCR) when LPSPI is busy, to support interfacing with different slave devices at different frequencies, you can change the TCR[PRESCALE] selection between SPI bus transfers by using Transmit Command (TCR).

NOTE

The minimum value shown in Table 564 is the minimum counter value, but the values of Clock Configuration (CCR) must also satisfy the data sheet specs based on the LPSPI functional clock frequency and prescaler value.

Table 564. Timing parameters

Clock Configuration (CCR) Clock Configuration 1 (CCR1)		Description	Minimum value	Maximum value
Field	Name			
SCKSET	SCK setup phase	Configures the SCK setup phase to (SCKSET + 1) cycles. The setup phase is the SCK high period when either CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. Otherwise, it is the SCK low period. The SCK period is defined as (SCKSET + SCKHLD + 2) and the duty cycle is the difference between SCKSET and SCKHLD.	0 (1 cycle)	255 (256 cycles)
SCKHLD	SCK hold phase	Configures the SCK hold phase to (SCKHLD + 1) cycles. The hold phase is the SCK low period when either CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. Otherwise, it is the SCK high period. The SCK period is defined as (SCKSET + SCKHLD + 2) and the duty cycle is the difference between SCKSET and SCKHLD.	0 (1 cycle)	255 (256 cycles)
PCSPCS	PCS-to-PCS delay	Configures the minimum delay between PCS negation and the next PCS assertion to (PCSPCS + PCSPCS + 2) cycles. When the command word is updated between transfers, there is a minimum of (PCSPCS + 1) cycles between the command word update and any change on PCS pins.	0 (2 cycles)	255 (512 cycles)
SCKSCK	SCK-to-SCK delay	Configures the delay during a continuous transfer between the last SCK edge of a frame and the first SCK edge of the continuing frame to (SCKSCK + 1) cycles. This is useful when the external slave requires a large delay between different words of an SPI bus transfer.	0 (1 cycle)	255 (256 cycles)

Table continues on the next page...

Table 564. Timing parameters (continued)

Clock Configuration (CCR) Clock Configuration 1 (CCR1)		Description	Minimum value	Maximum value
Field	Name			
PCSSCK	PCS-to-SCK delay	Configures the minimum delay between PCS assertion and the first SCK edge to (PCSSCK + 1) cycles.	0 (1 cycle)	255 (256 cycles)
SCKPCS	SCK-to-PCS delay	Configures the minimum delay between the last SCK edge and the PCS negation to (SCKPCS + 1) cycles.	0 (1 cycle)	255 (256 cycles)

Figure 379 shows the timing settings controlled by:

- TCR[CPHA]
- TCR[CPOL]
- CCR[SCKPCS]
- CCR[PCSSCK]
- CCR1[SCKSET]
- CCR1[SCKHLD]

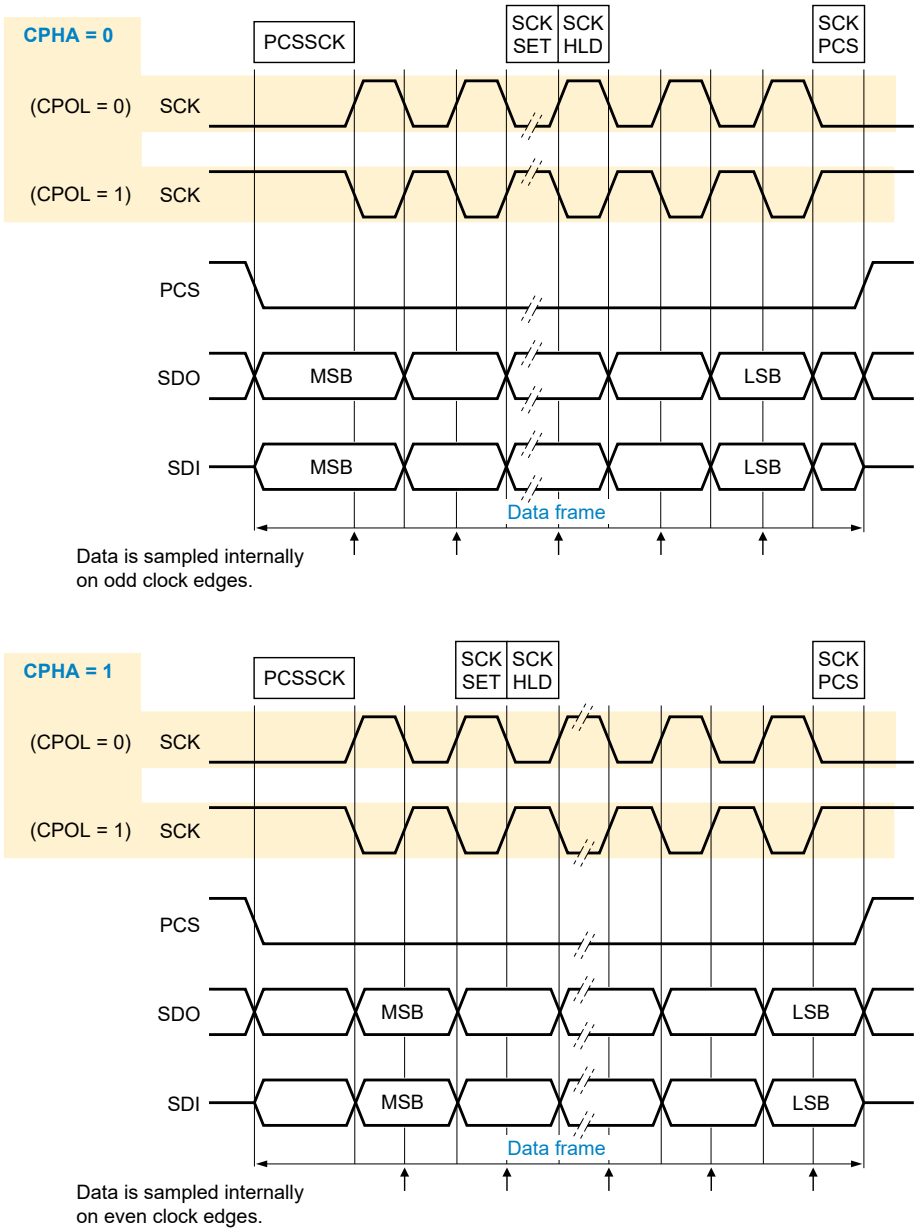


Figure 379. Clock phase (TCR[CPHA]) timing diagram example

To configure for a baud rate of 10 MHz with 50/50 duty cycle and with a functional clock frequency of 100 MHz, use the following settings:

- CCR1[SCKSET] = 0x4 (5 cycles)
- CCR1[SCKHLD] = 0x4 (5 cycles)
- CCR1[PCSPCS] = 0x8 (10 cycles)
- CCR1[SCKSCK] = 0x4 (5 cycles)
- CRR[PCSSCK] = 0x4 (5 cycles)
- CRR[SCKPCS] = 0x4 (5 cycles)
- TCR[PRESCALE] = 0x0 (divide by 1)

66.4.2.1.4 Pin configuration

Following are the pin configuration settings for half-duplex transfers:

- To swap directions or to support half-duplex transfers on the same pin, you can configure the SIN and SOUT pins using CFGR1[PINCFG].
- To specify whether an output data pin (SOUT, for example) 3-states when PCS is negated, or if the output data pin retains the last value, use CFGR1[OUTCFG].
- When configuring half-duplex transfers, you must configure the output data pins to 3-state when PCS is negated (CFGR1[OUTCFG] = 1).
- When performing half-duplex 2-bit transfers, you can write any value to CFGR1[PCSCFG].
- When performing half-duplex 4-bit transfers, you must write 1h to CFGR1[PCSCFG].

66.4.2.1.5 Clock loopback

Configure the LPSPI master to use one of the following clocks to sample the input data:

- The SCK output clock
- A delayed version of the SCK output clock

The delayed version of the SCK is chosen by the SCK pin output delay, plus the SCK pin input delay, and is selected by writing 1 to CFGR1[SAMPLE]. Enabling the loopback version of the SCK pin can improve the setup time of the input data from the slave.

See the chip data sheet for the specific input setup time in Master Loopback mode.

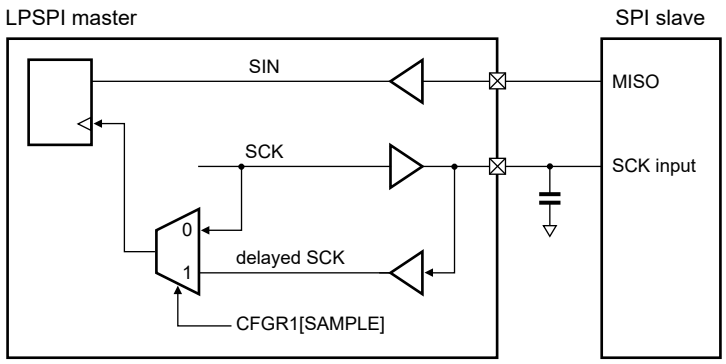


Figure 380. Clock loopback

66.4.2.2 Slave mode

LPSPI Slave mode:

- Uses the same shift register and logic that Master mode uses.
- Does not use Clock Configuration (CCR).
- Requires Transmit Command (TCR) to remain static (unchanged) during SPI bus transfers.

66.4.2.2.1 Transmit and command FIFO commands

You must initialize Transmit Command (TCR) before enabling LPSPI in Slave mode, although this register is not updated until after LPSPI is enabled. After LPSPI is enabled, you must make changes to this register only when LPSPI is idle. In Slave mode, the LPSPI command word in this register controls SPI attributes. Before the PCS input asserts, the transmit FIFO must be filled with transmit data, or the transmit error flag sets.

Table 565. Command word in Slave mode

Transmit Command (TCR)		Description
Field	Name	
CPOL	Clock polarity	Specifies the polarity of the external SCK input.
CPHA	Clock phase	Specifies the clock phase of transfer.
PRESCALE	Prescaler value	Specifies the LPSPI functional clock prescaler.
PCS	Peripheral chip select	Specifies which PCS is used. The polarity of PCS is static and configured by CFGR1[PCSPOL]. If CFGR1[PCSCFG] is not equal to zero, then do not select the PCS[3:2] pins.
LSBF	LSB first	Specifies whether LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first.
BYSW	Byte swap	Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing with devices that organize data as big-endian.
CONT	Continuous transfer	When continuous transfer is selected in Slave mode, after the number of bits indicated by FRAMESZ are transferred, LPSPI passes through and transmits the received data until the next PCS negation. Whatever is shifted in on the receive data is shifted out as transmit data considering that there is a 32-bit shift register.
CONTC	Continuing command	When the continuing command is enabled in Slave mode, after the number of bits indicated by FRAMESZ are transferred, RXMSK is considered equal to 1 and TXMSK is considered equal to 0 until the next PCS negation. CONTC can be used to change the direction of a transfer after the number of bits indicated by FRAMESZ.
RXMSK	Receive data mask	Masks the receive data; LPSPI does not store masked receive data in the receive FIFO or perform receive data matching. This option is useful for half-duplex transfers or to specify which fields are compared during receive data matching.
TXMSK	Transmit data mask	Masks the transmit data so that the masked transmit data is not pulled from transmit FIFO, and the output data pin is 3-stated (unless otherwise specified in CFGR1[OUTCFG]). This option is useful for half-duplex transfers.
WIDTH	Transfer width	Specifies the number of bits shifted on each SCK pulse: • 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats. • 2-bit and 4-bit half-duplex transfers are useful for interfacing with QuadSPI memory devices, and at least either TCR[TXMSK] or TCR[RXMSK] must be 1.
FRAMESZ	Frame size	Specifies the frame size in number of bits equal to (FRAMESZ + 1): • The minimum frame size is 8 bits. • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32 bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately. • If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO contain the remainder bits. For example, a 72-bit transfer consists of three words: the first and second words are 32 bits, and the third word is 8 bits.

66.4.2.2.2 Receive FIFO and data match

The receive FIFO stores receive data during SPI bus transfers. When TCR[RXMSK] = 1, the received data is discarded instead of storing the received data in the receive FIFO.

Receive data supports a receive data match function that can match received data against one of the two words in DMR0 and DMR1 or against a masked data word. You can also configure the data match function to compare only the first one or two received data words since the start of the frame:

- Received data that is already discarded because TCR[RXMSK] = 1 cannot cause the data match to set, and delays the match on the first received data word, until all discarded data is received.
- By using CFGR0[RDMO], you can also configure the receiver match function to discard all received data until a data match is detected.
- After a receive data match, to allow all subsequent data to be received, first write 0 to CFGR0[RDMO], then clear SR[DMF].

66.4.2.2.3 Partial received word

When the PCS pin deasserts and the receive shift register shifts in a partial word, you can configure the receive shift register to either discard the partial word or to store it in the receive FIFO. You must specify this using CFGR1[PARTIAL].

A partial word is defined as less than TCR[FRAMESZ] bits (when TCR[FRAMESZ] is equal or less than 32 bits, or it is the last word in a multi-word frame) or less than 32 bits (when TCR[FRAMESZ] is greater than 32 bits and not the last word in a multi-word frame).

A single-bit frame is not supported. A partial received word of 1 bit is supported, but a partial received frame of 1 bit is not supported.

66.4.2.2.4 Clocked interface

LPSPI supports interfacing with external masters that provide only clock and data pins (PCS is not required). This interface requires:

- Writing 1 to TCR[CPHA] (data is changed on the leading edge of SCK and captured on the following edge).
- Configuring the PCS input to be always asserted (CFGR1[PCSPOL*n*] = 1). For example, to configure PCS[0] to be always asserted, write 1 to PCSPOL[0], and do not configure PCS[0] in the pin muxing. The chip-level drives PCS to a certain value (ideally 1); you could use CFGR1[PCSPOL*n*] to invert that value.
- Writing 1 to CFGR1[AUTOPCS] to enable automatic PCS generation. When CFGR1[AUTOPCS] = 1, a minimum of four LPSPI functional clock cycles (divided by the selection specified in TCR[PRESCALE]) is required between the last SCK edge of one word and the first SCK edge of the next word.

66.4.2.3 Low-power modes

Table 566. Low-power modes

Chip mode	LPSPI operation
Run	Normal operation
Deep Sleep	Can continue operating in Deep Sleep mode if LPSPI is using an external or internal clock source that remains operating during Deep Sleep mode
Low-Power-Stop (also called Power Down)	Waits for the current transfer to complete any pending operation, before entering Low-Power-Stop mode

66.4.2.4 Debug mode

Table 567. Debug mode

Chip mode	LPSPI operation
Debug (the core is in Debug or Halted mode)	Can continue operating in Debug mode, if CR[DBGEN] = 1

66.4.2.5 Clocking

Table 568. LPSPI clocks

Type of clock	Description
Functional	• Asynchronous to the bus clock. • If the LPSPI functional clock remains enabled in low-power modes, then LPSPI can perform SPI bus transfers and low-power wakeups in both Master and Slave modes. • LPSPI divides the functional clock by a prescaler; the resulting frequency must be at least two times faster than the SPI external clock frequency (SCK).
External	• The LPSPI shift register is clocked directly by the SCK clock. • How the SCK clock is generated or supplied depends on the mode (Master or Slave): — In Master mode, the SCK clock is generated internally. — In Slave mode, the SCK clock is supplied externally.
Bus	The bus clock is only used for bus accesses to the LPSPI control and configuration registers. The bus clock frequency must be high enough to support the data bandwidth requirements of the LPSPI registers, including the FIFOs.

See the chip-specific LPSPI information for more.

66.4.2.6 Reset

Table 569. LPSPI resets

Type of reset	Description
Chip	Resets the LPSPI logic and registers to their default states.
Software	• Resets the LPSPI logic and registers to their default states, except for the Control register. • The LPSPI software reset is controlled using CR[RST].
FIFO	• Resets the transmit and command FIFO and the receive FIFO. • CR[RTF] and CR[RRF] are write-only. • After being reset, FIFO is empty.

66.4.2.7 Interrupts and DMA requests

The following table lists Slave mode sources (status flags) that can generate LPSPI interrupts and LPSPI slave transmit and receive DMA requests.

Table 570. Interrupts and DMA requests

Status (SR)		Description	Can generate		
Status flag	Name		Interrupt?	DMA request?	Low-power wake-up?
TDF	Transmit data flag	Indicates that data can be written to transmit FIFO, as configured by the transmit FIFO watermark, FCR[TXWATER].	Y	TX	Y
RDF	Receive data flag	Indicates that data can be read from the receive FIFO, as configured by the receive FIFO watermark, FCR[RXWATER].	Y	RX	Y
WCF	Word complete flag	Indicates that the word is complete and the last bit of the word has been sampled.	Y	N	Y
FCF	Frame complete flag	Indicates that the frame is complete and PCS is deasserted.	Y	RX	Y
TCF	Transfer complete flag	Indicates that transfer is complete, PCS is deasserted, and the transmit and command FIFO is empty.	Y	N	Y
TEF	Transmit error flag	Indicates a transmit and command FIFO underrun. In Master mode, when CFGR1[NOSTALL] = 0 (transfers stall when transmit FIFO is empty), TEF cannot be set.	Y	N	Y
REF	Receive error flag	Indicates a receive FIFO overflow. In Master mode, when CFGR1[NOSTALL] = 0 (transfers stall when receive FIFO is full), REF cannot be set.	Y	N	Y
DMF	Data match flag	Indicates that the received data matches the configured data match value.	Y	N	Y
MBF	Module busy flag	Indicates that LPSPI is busy performing an SPI bus transfer.	N	N	N

66.4.2.7.1 End-of-packet DMA transfer

The end-of-packet functionality is designed for serial interfaces where you may not be aware of the size of the transfer in advance and the data is pushed by an external device. Examples include UART receive, I2C Slave mode, and SPI Slave mode. The end-of-packet processing is intended to ensure that data is not stranded in either the receive FIFO or the DMA receive buffer. Support for end-of-packet processing must be implemented in both the serial interfaces and the DMA controller.

The condition that signals the end of packet is different for each serial interface but the serial peripheral and DMA process it in the same way. For example, UART end of packet is signaled by an idle line condition, I2C end of packet by a stop and/or repeated start condition, and SPI end of packet by PCS negation.

If you configure the serial peripheral to signal the end-of-packet condition to the DMA and the serial interface detects an end-of-packet condition, it asserts the DMA request for the receive FIFO irrespective of the watermark configuration. For larger watermark configurations, this ensures that the last few words of the transfer are first flushed from the receive FIFO.

The DMA then reads the contents of the receive FIFO, depending on the first word in the FIFO:

- If the receive FIFO is empty, the serial interface signals an end-of-packet condition to the DMA controller.
- If the receive FIFO is not empty, but the first word in the FIFO is the start of a new packet, then data is not pulled from the receive FIFO and the serial interface signals an end-of-packet condition to the DMA controller.

- If the receive FIFO is not empty, and the first word in the FIFO is not the start of a new packet, the DMA transfers the receive data as normal.

Because the DMA may be transferring multiple words on each request, the end-of-packet condition persists until the DMA minor loop is complete and no additional data is pulled from the receive FIFO. The status flag that triggered the end-of-packet condition is cleared when the minor loop completes following end of packet being signaled to the DMA controller.

When the DMA detects the end-of-packet condition, it writes all received words up to the end of the packet into the system memory and saves the destination address for the word after the last valid data. The DMA then terminates the channel as if the major loop is complete, including final offsets and optional interrupts, channel linking and scatter/gather. The final destination address can optionally be saved in the system memory or is available in the destination address register.

After the DMA terminates the major loop, no servicing of the receive FIFO occurs until either software or hardware reconfigures the DMA (for example, channel linking or scatter/gather). This delay must be minimized to avoid receiver FIFO overrun. NXP does not recommend automatic DMA end-of-packet processing when there are only a few words transferred between end-of-packet conditions. That is because the DMA spends more time processing the end of packet than transferring the data. For example, the UART idle line length must be increased as needed to avoid an excessive number of idle conditions.

66.4.2.8 Peripheral triggers

The connection of the LPSPI peripheral triggers with other peripherals depends on the device that is used.

Table 571. Peripheral triggers

Type of trigger	Description	Additional information
Frame output	The frame output trigger: • Asserts at the end of each frame (when PCS deasserts). • Remains asserted for one cycle of the LPSPI functional clock divided by the configuration defined in TCR[PRESCALE].	LPSPI generates two output triggers that can be connected to other peripherals on the chip.
Word output	The word output trigger: • Asserts at the end of each received word. • Remains asserted for one cycle of the LPSPI functional clock divided by the configuration defined in TCR[PRESCALE].	
Input	To control the start of an LPSPI bus transfer, the LPSPI input trigger can be selected instead of the HREQ input: • The LPSPI input trigger is synchronized, and must assert for at least two cycles of the LPSPI functional clock divided by the configuration defined in TCR[PRESCALE] so that the input trigger can be detected. • When LPSPI is busy, the HREQ input (and therefore the LPSPI input trigger) is ignored. • When LPSPI is busy, both the HREQ and LPSPI input triggers are ignored. They are used to start a new transfer when LPSPI is idle.	

66.4.3 External signals

Table 572. External signals

Signal	Name	Description	I/O
SCK	Serial clock	• Input in Slave mode	I/O

Table continues on the next page...

Table 572. External signals (continued)

Signal	Name	Description	I/O
		• Output in Master mode	
PCS[0]	Peripheral chip select	• Input in Slave mode • Output in Master mode	I/O
PCS[1]/HREQ	Peripheral chip select or host request	Host request pin is selected when CFGR0[HREN] = 1 and CFGR0[HRSEL] = 0: • Input in either Slave mode or when used as master host request • Output in either Master mode or when used as slave host request	I/O
PCS[2]/DATA[2]	Peripheral chip select or data pin 2 during parallel data transfers	When CFGR1[PCSCFG] = 0: • Input in Slave mode • Output in Master mode When CFGR1[PCSCFG] = 1: • Input in half-duplex parallel data receive transfers • Output in half-duplex parallel data transmit transfers	I/O
PCS[3]/DATA[3]	Peripheral chip select or data pin 3 during parallel data transfers	When CFGR1[PCSCFG] = 0: • Input in Slave mode • Output in Master mode When CFGR1[PCSCFG] = 1: • Input in half-duplex parallel data receive transfers • Output in half-duplex parallel data transmit transfers	I/O
SOUT/DATA[0]	Serial data output	Can be configured as serial data input signal (used as data pin 0 in half-duplex parallel data transfers)	I/O
SIN/DATA[1]	Serial data input	Can be configured as serial data output signal (used as data pin 1 in half-duplex parallel data transfers)	I/O

66.4.4 Initialization

This module does not require initialization.

66.4.5 Memory map and registers

NOTE

- Writing to a read-only register or reading a write-only register can cause bus errors.
- LPSPI does not check values programmed in registers for validity, so you must take care to write valid values only.

66.4.5.1 LPSPI register descriptions

LPSPI provides an efficient interface to an SPI bus, either as a master or slave. An SPI bus is a synchronous serial communication interface used in embedded systems. It is typically used to perform short distance communications between microcontrollers and peripheral devices, on printed circuit boards. Typical applications include interfacing with secure digital cards and LCD displays.

66.4.5.1.1 LPSPI memory map

LPSPI0 base address: 4009_2000h

LPSPI1 base address: 4009_3000h

LPSPI2 base address: 4009_4000h

LPSPI3 base address: 4009_5000h

LPSPI4 base address: 400B_4000h

LPSPI5 base address: 400B_5000h

LPSPI6 base address: 400B_6000h

LPSPI7 base address: 400B_7000h

LPSPI8 base address: 400B_8000h

LPSPI9 base address: 400B_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0200_0004h
4h	Parameter (PARAM)	32	R	0004_0303h
10h	Control (CR)	32	RW	0000_0000h
14h	Status (SR)	32	RW	0000_0001h
18h	Interrupt Enable (IER)	32	RW	0000_0000h
1Ch	DMA Enable (DER)	32	RW	0000_0000h
20h	Configuration 0 (CFGR0)	32	RW	0000_0000h
24h	Configuration 1 (CFGR1)	32	RW	0000_0000h
30h	Data Match 0 (DMR0)	32	RW	0000_0000h
34h	Data Match 1 (DMR1)	32	RW	0000_0000h
40h	Clock Configuration (CCR)	32	RW	0000_0000h
44h	Clock Configuration 1 (CCR1)	32	RW	0000_0000h
58h	FIFO Control (FCR)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
5Ch	FIFO Status (FSR)	32	R	0000_0000h
60h	Transmit Command (TCR)	32	RW	0000_001Fh
64h	Transmit Data (TDR)	32	W	0000_0000h
70h	Receive Status (RSR)	32	R	0000_0002h
74h	Receive Data (RDR)	32	R	0000_0000h
78h	Receive Data Read Only (RDROR)	32	R	0000_0000h
3FCh	Transmit Command Burst (TCBR)	32	W	0000_0000h
400h - 5FCh	Transmit Data Burst (TDBR0 - TDBR127)	32	W	0000_0000h
600h - 7FCh	Receive Data Burst (RDBR0 - RDBR127)	32	R	0000_0000h

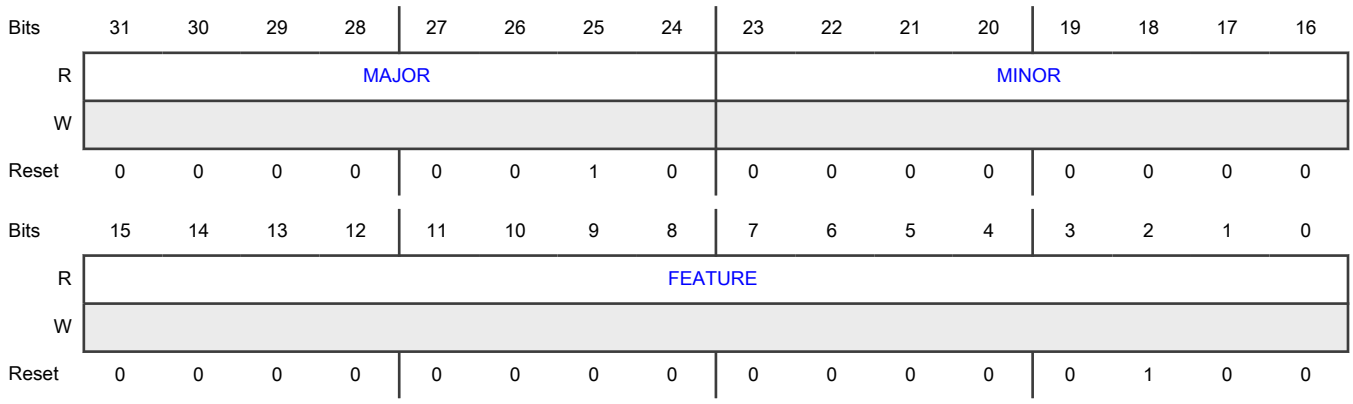
66.4.5.1.2 Version ID (VERID)

Offset

Register	Offset
VERID	0h

Function
Contains version numbers for the module design and feature set.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Indicates the major version number of the module specification.
23-16 MINOR	Minor Version Number Indicates the minor version number of the module specification.
15-0 FEATURE	Module Identification Number Indicates the feature set number 0000_0000_0000_0100b - Standard feature set supporting a 32-bit shift register. All other values are reserved.

66.4.5.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function

Contains:

- Number of PCS pins.
- Receive FIFO size.
- Transmit FIFO size.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								PCSNUM							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXFIFO								TXFIFO							
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

Fields

Field	Function
31-24 —	Reserved
23-16 PCSNUM	PCS Number Indicates the number of PCS pins supported.
15-8 RXFIFO	Receive FIFO Size Indicates the maximum number of words in the receive FIFO. The maximum number of words is 2^{RXFIFO}.
7-0 TXFIFO	Transmit FIFO Size Indicates the maximum number of words in the transmit FIFO. The maximum number of words is 2^{TXFIFO}.

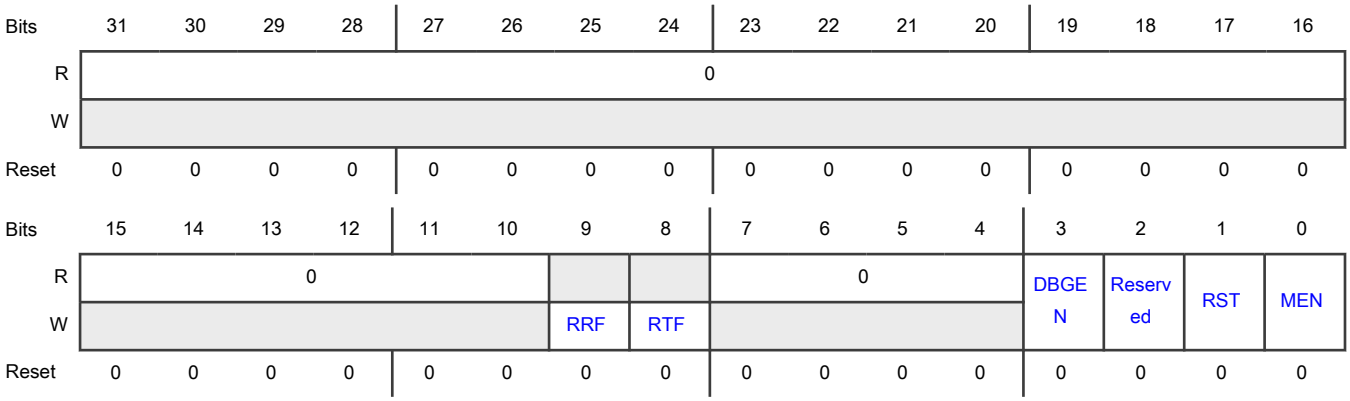
66.4.5.1.4 Control (CR)

Offset

Register	Offset
CR	10h

Function
Contains fields that control the module operation.

Diagram



Fields

Field	Function
31-10	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
9 RRF	Reset Receive FIFO Deletes all entries in the receive FIFO. This field always reads 0. 0b - No effect 1b - Reset
8 RTF	Reset Transmit FIFO Deletes all entries in the transmit FIFO. This field always reads 0. 0b - No effect 1b - Reset
7-4 —	Reserved
3 DBGEN	Debug Enable Enables LPSPI when the CPU is in Debug mode. If this field is 0, LPSPI is disabled when the CPU is halted; the PCS pin is deasserted after the transmit FIFO is empty regardless of the state of Transmit Command (TCR). You must update this field only when LPSPI is disabled (MEN = 0). 0b - Disable 1b - Enable
2 —	Reserved
1 RST	Software Reset Resets all internal logic and registers, except Control (CR). The reset takes effect immediately and remains asserted until you write 0 to it. There is no minimum delay required before clearing the software reset by writing 0. 0b - Not reset 1b - Reset
0 MEN	Module Enable Enables the module. After writing 0, MEN remains set until LPSPI has completed the current transfer and is idle. 0b - Disable 1b - Enable

66.4.5.1.5 Status (SR)

Offset

Register	Offset
SR	14h

Function
Contains data flow status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							MBF	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		DMF	REF	TEF	TCF	FCF	WCF	0						RDF	TDF
W			W1C	W1C	W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Function
31-25 —	Reserved
24 MBF	Module Busy Flag Indicates, in Master mode, whether there is data to transmit and LPSPI is able to transmit (for example, the HREQ pin is asserted). The HREQ pin deasserts after the PCS pin deasserts and the LPSPI master has waited for half the time specified in CCR[DBT] with no new data to transmit. Slave mode sets this flag when LPSPI is enabled and PCS is asserted. 0b - LPSPI is idle 1b - LPSPI is busy
23-14 —	Reserved
13 DMF	Data Match Flag Indicates whether the received data matches DMR0[MATCH0] and/or DMR1[MATCH1] (as configured by CFGR1[MATCFG]).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - No match 1b - Match When writing 0b - No effect 1b - Clear the flag
12 REF	Receive Error Flag Indicates a receive FIFO overflow error. When this flag is set: 1. End the transfer. 2. Empty the receive FIFO. 3. Clear this flag. 4. Restart the transfer from the beginning. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No overflow 1b - Overflow When writing 0b - No effect 1b - Clear the flag
11 TEF	Transmit Error Flag Indicates a transmit FIFO underrun error. When this flag is set: 1. End the transfer. 2. Clear this flag. 3. Restart the transfer from the beginning. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No underrun 1b - Underrun

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When writing 0b - No effect 1b - Clear the flag
10 TCF	Transfer Complete Flag Indicates, in Master mode, whether all transfers are complete and LPSPI has returned to the Idle state and the transmit FIFO is empty. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not complete 1b - Complete When writing 0b - No effect 1b - Clear the flag
9 FCF	Frame Complete Flag Indicates whether a frame transfer is complete after PCS deasserts. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not complete 1b - Complete When writing 0b - No effect 1b - Clear the flag
8 WCF	Word Complete Flag Indicates whether the last bit of a received word is sampled. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not complete 1b - Complete When writing

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Clear the flag
7-2 —	Reserved
1 RDF	Receive Data Flag Indicates whether the number of words in the receive FIFO is greater than the value in FCR[RXWATER]. 0b - Receive data not ready 1b - Receive data ready
0 TDF	Transmit Data Flag Indicates whether the number of words in the transmit FIFO is equal to or less than the value in FCR[TXWATER]. 0b - Transmit data not requested 1b - Transmit data requested

66.4.5.1.6 Interrupt Enable (IER)

Offset

Register	Offset
IER	18h

Function
Enables interrupts based on data flow and errors.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		DMIE	REIE	TEIE	TCIE	FCIE	WCIE	0						RDIE	TDIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-14 —	Reserved
13 DMIE	Data Match Interrupt Enable Enables the data match interrupt. 0b - Disable 1b - Enable
12 REIE	Receive Error Interrupt Enable Enables the receive complete interrupt. 0b - Disable 1b - Enable
11 TEIE	Transmit Error Interrupt Enable Enables the transmit complete interrupt. 0b - Disable 1b - Enable
10 TCIE	Transfer Complete Interrupt Enable Enables the transfer complete interrupt. 0b - Disable 1b - Enable
9 FCIE	Frame Complete Interrupt Enable Enables the frame complete interrupt. 0b - Disable 1b - Enable
8 WCIE	Word Complete Interrupt Enable Enables the word complete interrupt. 0b - Disable 1b - Enable
7-2 —	Reserved
1 RDIE	Receive Data Interrupt Enable Enables the receive data interrupt.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
0 TDIE	Transmit Data Interrupt Enable Enables the transmit data interrupt. 0b - Disable 1b - Enable

66.4.5.1.7 DMA Enable (DER)

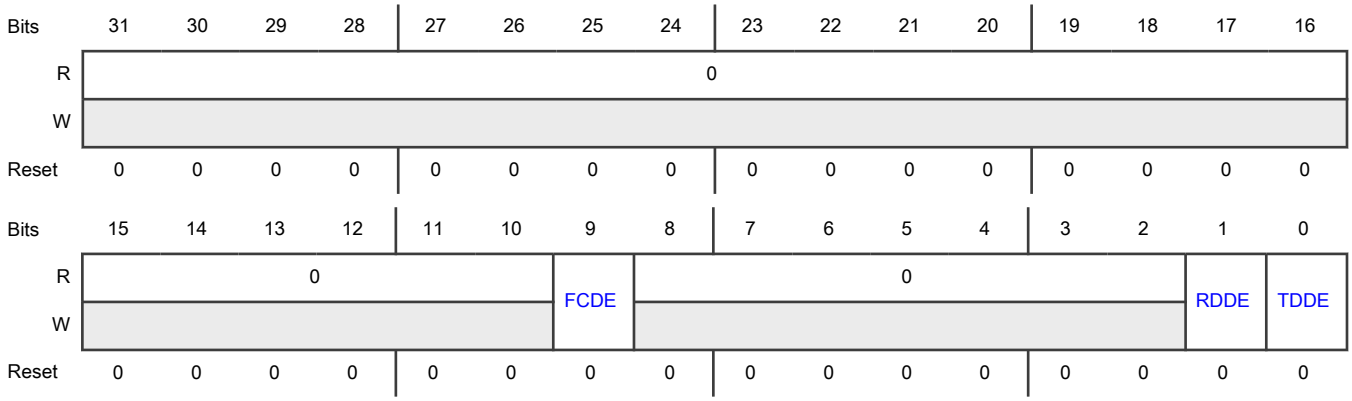
Offset

Register	Offset
DER	1Ch

Function

Enables the DMA data flow.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 FCDE	Frame Complete DMA Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Enables DMA end-of-packet processing. After the last word of a frame is read from the receive data FIFO, reading the receive data FIFO returns an end-of-packet signal with the receive data forced to FFFF_FFFFh. This continues until the DMA minor loop completes, and then SR[FCF] deasserts if the receive FIFO is empty or if LPSPI is busy (SR[MBF] = 1). 0b - Disable 1b - Enable
8-2 —	Reserved
1 RDDE	Receive Data DMA Enable Enables the receive data DMA. 0b - Disable 1b - Enable
0 TDDE	Transmit Data DMA Enable Enables the transmit data DMA. 0b - Disable 1b - Enable

66.4.5.1.8 Configuration 0 (CFGR0)

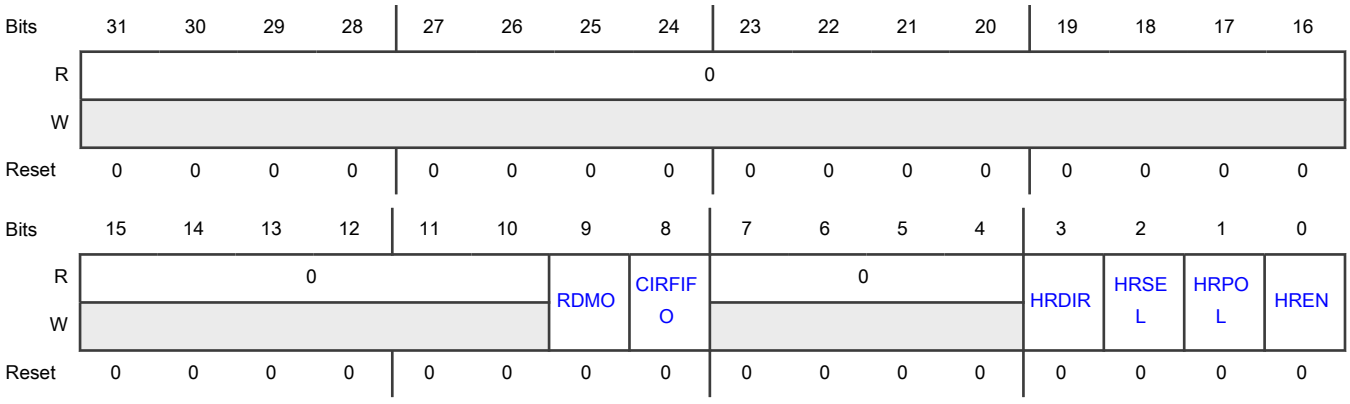
Offset

Register	Offset
CFGR0	20h

Function

Includes fields to configure LPSPI.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 RDMO	Receive Data Match Only Enables receive data match. When enabled, all received data that does not cause SR[DMF] to assert is discarded: • Write 1 to this field when LPSPI is idle and SR[DMF] = 0. • After SR[DMF] = 1, this field is ignored. • To ensure that no receive data is lost when disabling RDMO, write 0 to this field before clearing SR[DMF]. See CFGR1[MATCFG] for the received data matching options. When disabled, all received data is stored in the receive FIFO. 0b - Disable 1b - Enable
8 CIRFIFO	Circular FIFO Enable Enables circular FIFO. When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO is emptied as in normal operation, but when LPSPI is idle and the transmit FIFO is empty, the read pointer value is restored from the temporary register. This restoring of the read pointer causes the contents of the transmit FIFO to be cycled through repeatedly. **NOTE** The read pointer is restored for as long as this field is 1. Writing additional words to the FIFO when this field is 1 adds them to the end of the FIFO, up to the size of the transmit FIFO. 0b - Disable 1b - Enable
7-4 —	Reserved
3 HRDIR	Host Request Direction Specifies the direction of the HREQ pin. You must configure the HREQ pin only as an output when LPSPI is in Slave mode. The HREQ pin direction must be an input for Master mode. 0b - Input 1b - Output
2 HRSEL	Host Request Select Specifies the source of the host request input. When the host request function is enabled with the HREQ pin, the PCS[1] function is disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - HREQ pin 1b - Input trigger
1 HRPOL	Host Request Polarity Specifies the polarity of the HREQ pin or input trigger. 0b - Active high 1b - Active low
0 HREN	Host Request Enable Enables LPSPI, in Master mode, to start a new SPI bus transfer only if the host request input is asserted. When LPSPI is busy, the host request input is ignored. In Slave mode, causes the HREQ output pin to assert when data is available to be transmitted. 0b - Disable 1b - Enable

66.4.5.1.9 Configuration 1 (CFGR1)

Offset

Register	Offset
CFGR1	24h

Function

Includes fields to configure LPSPI. You must write to this register only when LPSPI is disabled.

In addition to pin and output configurations, this register contains match configuration details; the following table shows match conditions specified in MATCFG.

Table 573. Match conditions for CFGR1[MATCFG]

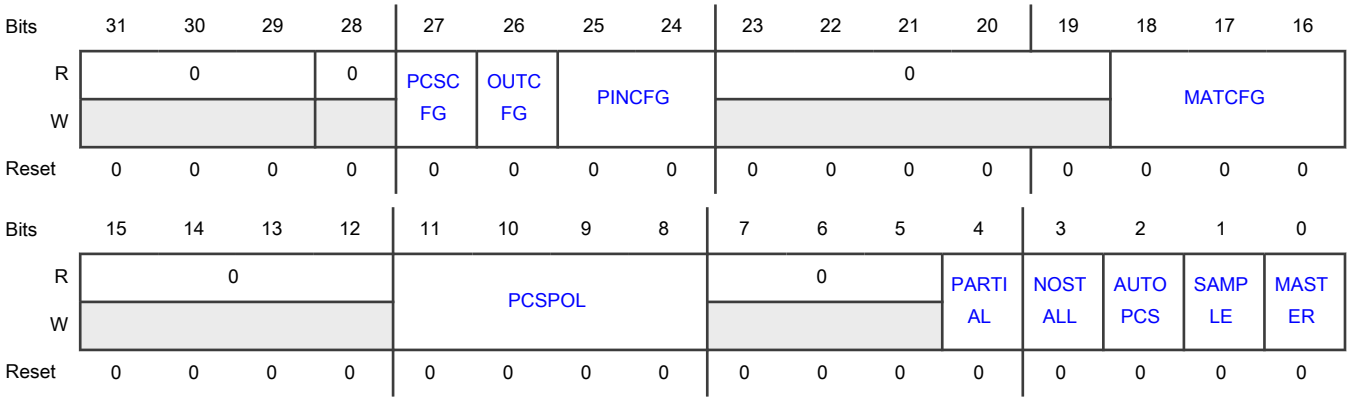
Condition	Description
Match first data word with compare word	Match if first data word equals MATCH0 logically ORed with MATCH1 first_data_word == (MATCH0 || MATCH1)
Match any data word with compare word	Match if any data word equals MATCH0 logically ORed with MATCH1 any_data_word == (MATCH0 || MATCH1)
Sequential match, first data word	Match if first data word equals MATCH0, and second data word equals MATCH1 (first_data_word == MATCH0) && (second_data_word == MATCH1)
Sequential match, any data word	Match if any data word equals MATCH0, and the next data word equals MATCH1

Table continues on the next page...

Table 573. Match conditions for CFGR1[MATCFG] (continued)

Condition	Description
	(any_data_word == MATCH0) && (next_data_word == MATCH1)
Match first data word (masked) with compare word (masked)	Match if first data word logically ANDed with MATCH1 equals MATCH0 logically ANDed with MATCH1 (first_data_word && MATCH1) == (MATCH0 && MATCH1)
Match any data word (masked) with compare word (masked)	Match if any data word logically ANDed with MATCH1 equals MATCH0 logically ANDed with MATCH1 (any_data_word && MATCH1) == (MATCH0 && MATCH1)

Diagram



Fields

Field	Function
31-29 —	Reserved
28 —	Reserved
27 PCSCFG	Peripheral Chip Select Configuration Specifies PCS pin configuration. When performing parallel transfers, you must configure this field to enable the desired transfer. 0b - PCS[3:2] configured for chip select function 1b - PCS[3:2] configured for half-duplex 4-bit transfers (PCS[3:2] = DATA[3:2])
26 OUTCFG	Output Configuration Specifies whether the output data is 3-stated between accesses (when PCS is deasserted). When performing half-duplex transfers, this field must be 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Retain last value 1b - 3-stated
25-24 PINCFG	Pin Configuration Specifies the pins used for input and output data during serial transfers. This field is ignored when performing parallel transfers. 00b - SIN is used for input data; SOUT is used for output data 01b - SIN is used for both input and output data; only half-duplex serial transfers are supported 10b - SOUT is used for both input and output data; only half-duplex serial transfers are supported 11b - SOUT is used for input data; SIN is used for output data
23-19 —	Reserved
18-16 MATCFG	Match Configuration Specifies the condition that causes SR[DMF] to assert. See the match conditions listed in Table 1 for more information. **NOTE** When writing to this field, either the old value or new value must be in the disabled state (0). You cannot transition from a nonzero value to another nonzero value. 000b - Match is disabled 001b - Reserved 010b - Match first data word with compare word 011b - Match any data word with compare word 100b - Sequential match, first data word 101b - Sequential match, any data word 110b - Match first data word (masked) with compare word (masked) 111b - Match any data word (masked) with compare word (masked)
15-12 —	Reserved
11-8 PCSPOL	Peripheral Chip Select Polarity Specifies the polarity of each PCS pin. Bit *n* in this field (the least-significant bit is bit 0) corresponds to PCS[*n*]. **NOTE** The entire PCSPOL field is not fully supported in every LPSPI module instance. See the LPSPI chip-specific information.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000b - Active low 0001b - Active high
7-5 —	Reserved
4 PARTIAL	Partial Enable Specifies whether LPSPI, when in Slave mode, stores a partial received word in the receive FIFO, or discards it, when PCS deasserts. See Partial received word for more information. 0b - Discard 1b - Store
3 NOSTALL	No Stall Disables a normal operating feature that causes LPSPI, when in Master mode, to stall transfers when the transmit FIFO is empty or when the receive FIFO is full. This feature prevents transmit FIFO underruns and receive FIFO overruns. Writing 1 to this field disables this functionality. 0b - Disable 1b - Enable
2 AUTOPCS	Automatic PCS Enables automatic PCS generation. For correct operation in Slave mode, LPSPI requires the PCS signal to deassert between frames. Writing 1 to this field generates an internal PCS signal at the end of each transfer word when TCR[CPHA] = 1. When this field is 1, SCK must remain idle for at least four LPSPI functional clock cycles, divided by the prescaler (see TCR[PRESCALE]) selected between each word to ensure correct operation. This field is ignored in Master mode. 0b - Disable 1b - Enable
1 SAMPLE	Sample Point Specifies the SCK clock edge on which LPSPI, when in Master mode, samples input data. Writing 1 to this field causes LPSPI to sample input data on a delayed loopback SCK clock edge, which improves the setup time when sampling data (see Clock loopback). In this configuration, the input data setup time in Master mode is equal to the input data setup time in Slave mode. In Slave mode, this field is ignored. **NOTE** When SAMPLE = 1, both the input and output buffers must be enabled for the SCK pin. 0b - SCK edge 1b - Delayed SCK edge

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 MASTER	Master Mode Specifies the LPSPI operating mode, Master or Slave. This field directly controls the direction of the SCK and PCS pins. 0b - Slave mode 1b - Master mode

66.4.5.1.10 Data Match 0 (DMR0)

Offset

Register	Offset
DMR0	30h

Function

Specifies the match data to be used when data matching is enabled. See CFGR1[MATCFG] for the received data matching options.

NOTE

Do not change the value in this register when CFGR1[MATCFG] > 0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MATCH0	Match 0 Value Specifies the MATCH0 value to be compared against received data.

66.4.5.1.11 Data Match 1 (DMR1)

Offset

Register	Offset
DMR1	34h

Function

Specifies the match data to be used when data matching is enabled. See CFGR1[MATCFG] for the received data matching options.

NOTE

Do not change the value in this register while CFGR1[MATCFG] > 0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCH1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MATCH1	Match 1 Value Specifies the MATCH1 value to be compared against received data.

66.4.5.1.12 Clock Configuration (CCR)

Offset

Register	Offset
CCR	40h

Function

Contains clock configuration fields that are used only in Master mode; you can only change them when LPSPI is disabled (CR[MEN] = 0).

Warning

Writing a 32-bit value to this register overwrites Clock Configuration 1 (CCR1); DBT and SCKDIV always read 0.

To avoid overwriting CCR1, do one of the following:

- Write to all four fields in Clock Configuration (CCR) simultaneously and only once in a 32-bit data.
- Modify the values of CCR[SCKPCS] and/or CCR[PCSSCK]; write only these two upper bytes in a 16-bit data or one of them in an 8-bit data.
- Modify CCR1[PCSPCS] and CCR1[SCKSCK] only or CCR1[SCKSET] and CCR1[SCKHLD] only, write respectively to CCR[DBT] or CCR[SCKDIV] in 8-bit data.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SCKPCS								PCSSCK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0							
W	DBT								SCKDIV							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 SCKPCS	SCK-to-PCS Delay Configures SCK-to-PCS delay. In Master mode, this field helps you configure the delay from the last SCK edge to PCS negation: • The delay is equal to (SCKPCS + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The minimum delay is one cycle. See Figure 379 for more information.
23-16 PCSSCK	PCS-to-SCK Delay Configures PCS-to-SCK delay. In Master mode, this field helps you configure the delay from PCS assertion to the first SCK edge: • The delay is equal to (PCSSCK + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The minimum delay is one cycle. See Figure 379 for more information.
15-8 DBT	Delay Between Transfers Configures the delay between transfers. Writing to this field updates the contents of CCR1[PCSPCS] and CCR1[SCKSCK].

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 SCKDIV	SCK Divider Updates the contents of CCR1[SCKSET] and CCR1[SCKHLD]. Baud rate = function clock ÷ (2^PRESCALE × (SCKSET + SCKHLD + 2))

66.4.5.1.13 Clock Configuration 1 (CCR1)

Offset

Register	Offset
CCR1	44h

Function

Contains clock configuration fields, which are used only in Master mode. You can change them only when LPSPI is disabled (CR[MEN] = 0).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SCKSCK								PCSPCS							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SCKHLD								SCKSET							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 SCKSCK	SCK Inter-Frame Delay Configures SCK inter-frame delay in Master mode: • This field helps you configure the delay from the last SCK pulse of a frame and the first SCK pulse of the following frame, in a continuous transfer. • The delay is equal to (SCKSCK + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The minimum delay is one cycle.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE For backward compatibility, writing to CCR[DBT] updates this field with the value written.
23-16 PCSPCS	PCS to PCS Delay Configures PCS to PCS delay in Master mode: • This field helps you configure the delay from the PCS negation to the next PCS assertion. • The delay is equal to (PCSPCS + PCSPCS + 2) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The minimum delay is two cycles. • Half of the delay (PCSPCS + 1) occurs before PCS assertion and the other half of the delay (PCSPCS + 1) occurs after PCS negation. If the command word is updated between two transfers, then the command word is updated halfway between the PCS negation of the last transfer and PCS assertion of the next transfer. • The command word specifies which PCS signal is used, the polarity and phase of the SCK signal, and the selected prescaler. **NOTE** For backward compatibility, writing to CCR[DBT] updates this field with (DBT÷2) rounded up.
15-8 SCKHLD	SCK Hold Configures the hold phase of the SCK pin in Master mode: • The hold phase is the delay between the SCK edge that samples the receive data and the SCK edge that drives the transmit data. • It is the SCK low period when CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. It is the SCK high period when CPHA = 0, CPOL = 0 and CPHA = 1, CPOL = 1. • The SCK hold phase delay is equal to (SCKHLD + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The minimum delay is one cycle. • The SCK period is equal to (SCKSET + SCKHLD + 2) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The SCK duty cycle is based on the difference between SCKSET and SCKHLD. You must configure both these fields to the same value for a 50/50 duty cycle. See Figure 379 for more information. **NOTE** For backward compatibility, writing to CCR[SCKDIV] updates this field with (SCKDIV ÷ 2) rounded down.
7-0 SCKSET	SCK Setup Configures the setup phase of the SCK pin in Master mode:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• The setup phase is the delay between the SCK edge that drives the transmit data and the SCK edge that samples the receive data. • It is the SCK high period when CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. It is the SCK low period when CPHA = 0 and CPOL = 0, or CPHA = 1 and CPOL = 1. • The SCK setup phase delay is equal to (SCKSET + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The minimum delay is one cycle. • The SCK period is equal to (SCKSET + SCKHLD + 2) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]). • The SCK duty cycle is based on the difference between SCKSET and SCKHLD. You must configure both these fields to the same value for a 50/50 duty cycle. See Figure 379 for more information. **NOTE** For backward compatibility, writing to CCR[SCKDIV] updates this field with (SCKDIV ÷ 2) rounded up.

66.4.5.1.14 FIFO Control (FCR)

Offset

Register	Offset
FCR	58h

Function

Contains the receive FIFO and transmit FIFO watermark values.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0				RXWATER			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0				TXWATER			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-19 —	Reserved
18-16 RXWATER	Receive FIFO Watermark Causes LPSPI to set SR[RDF] when the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size truncates the written value.
15-8 —	Reserved
7-3 —	Reserved
2-0 TXWATER	Transmit FIFO Watermark Causes LPSPI to set SR[TDF] when the number of words in the transmit FIFO is equal to or less than TXWATER. Writing a value equal to or greater than the FIFO size truncates the written value.

66.4.5.1.15 FIFO Status (FSR)

Offset

Register	Offset
FSR	5Ch

Function
Contains fields that indicate the number of words currently stored in the receive and transmit FIFOs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0				RXCOUNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0				TXCOUNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-20 —	Reserved
19-16 RXCOUNT	Receive FIFO Count Indicates the number of words currently stored in the receive FIFO.
15-8 —	Reserved
7-4 —	Reserved
3-0 TXCOUNT	Transmit FIFO Count Indicates the number of words currently stored in the transmit FIFO.

66.4.5.1.16 Transmit Command (TCR)

Offset

Register	Offset
TCR	60h

Function

Pushes the data into the transmit FIFO, in the same order as written.

When you write to either this register or to Transmit Data (TDR), each write pushes data into the transmit FIFO. You must write to this register only using 32-bit writes, which are tagged and cause the command register to update; after that the entry reaches the top of the FIFO and LPSPI is enabled. This allows changes to the command word and the transmit data itself to be interleaved. That is, writes to the two registers can be interleaved (write command word, then data word, then command word, and so on). Changing the command word causes all subsequent SPI bus transfers to be performed using the new command word:

- In Master mode, writing a new command word does not initiate a new transfer, unless TXMSK is 1. Transfers are initiated by transmit data in the transmit FIFO, or by a new command word (with TXMSK = 1). Hardware writes 0 to TXMSK when PCS deasserts.
- In Master mode, if the command word is changed before an existing frame has completed, then the existing frame terminates and the command word updates. The command word can be changed during a continuous transfer, if CONTC of the new command word is 1 and the command word is written on a frame size boundary.
- In Slave mode, the command word must be changed only when LPSPI is idle and there is no SPI bus transfer.

Avoid resetting the transmit FIFO after writing to this register; wait for the command register to update from the FIFO first.

Avoid register reading problems: Reading this register returns the current state of the register. Reading this register at the same time that it is loaded from the transmit FIFO can return an incorrect register value. It is recommended to:

- Read this register when the transmit FIFO is empty.
- Read this register more than once and then compare the returned values.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CPOL	CPHA	PRESCALE			Reserved	PCS		LSBF	BYSW	CONT	CONTC	RXMSK	TXMSK	WIDTH	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				FRAMESZ											
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Fields

Field	Function
31 CPOL	Clock Polarity Specifies the value of SCK when it is idle. You can update this field only when PCS is deasserted. See Figure 379 for more information. 0b - Inactive low 1b - Inactive high
30 CPHA	Clock Phase Indicates whether data is captured or changed on the leading edge of SCK and captured or changed on the following edge of SCK. You can update this field only when PCS is deasserted. See Figure 379 for more information. 0b - Captured 1b - Changed
29-27 PRESCALE	Prescaler Value Specifies the division of the LPSPI functional clock. For all SPI bus transfers, this value is applied to Clock Configuration (CCR). You can update this field only when PCS is deasserted. 000b - Divide by 1 001b - Divide by 2 010b - Divide by 4 011b - Divide by 8 100b - Divide by 16 101b - Divide by 32

Table continues on the next page...

Table continued from the previous page...

Field	Function
	110b - Divide by 64 111b - Divide by 128
26 —	Reserved
25-24 PCS	Peripheral Chip Select Configures the peripheral chip select used for the transfer. This field is updated only when PCS is deasserted. **NOTE** This entire field is not fully supported in every LPSPI module instance. See the chip-specific LPSPI information. 00b - Transfer using PCS[0] 01b - Transfer using PCS[1] 10b - Transfer using PCS[2] 11b - Transfer using PCS[3]
23 LSBF	LSB First Indicates whether data is transferred with MSB first or LSB first. 0b - MSB first 1b - LSB first
22 BYSW	Byte Swap Swaps the contents of [31:24] with [7:0] and [23:16] with [15:8] for each transmit data word read from the FIFO and for each received data word stored to the FIFO (or compared with match registers). 0b - Disable byte swap 1b - Enable byte swap
21 CONT	Continuous Transfer Enables continuous transfer: • In Master mode, this field keeps PCS asserted at the end of the frame size until a command word is received that starts a new frame. • In Slave mode, when this field is enabled, LPSPI only transmits the first FRAMESZ bits, after which LPSPI transmits received data (assuming a 32-bit shift register) until the next PCS negation. 0b - Disable 1b - Enable
20 CONTC	Continuing Command Enables the command word to be changed within a continuous transfer in Master mode:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• The initial command word must enable continuous transfer (CONT = 1). • The continuing command must have CONTC = 1. • The continuing command word must be loaded on a frame size boundary. For example, if the continuous transfer has a frame size of 64 bits, then a continuing command word must be loaded on a 64-bit boundary. In Slave mode, this field modifies the internal RXMSK and TXMSK configuration after the first FRAMESZ bits and until PCS negation: • Receive data is discarded after the first FRAMESZ bits. If CONT is also 1, this does not block the transmission of received data. • Transmit data is not masked after the first FRAMESZ bits. This allows the first FRAMESZ bits to be received and a response transmitted. 0b - Command word for start of new transfer 1b - Command word for continuing transfer
19 RXMSK	Receive Data Mask Masks receive data (receive data is not stored in the receive FIFO). 0b - Normal transfer 1b - Mask receive data
18 TXMSK	Transmit Data Mask Masks transmit data (no data is loaded from the transmit FIFO and the output pin is 3-stated). In Master mode, TXMSK initiates a new transfer that cannot be aborted by another command word. TXMSK automatically transitions to 0 at the end of the transfer. 0b - Normal transfer 1b - Mask transmit data
17-16 WIDTH	Transfer Width Configures serial (1-bit) or parallel transfers. For half-duplex parallel transfers, either RXMSK or TXMSK must be 1. 00b - 1-bit transfer 01b - 2-bit transfer 10b - 4-bit transfer 11b - Reserved
15-12 —	Reserved
11-0 FRAMESZ	Frame Size Configures the frame size in number of bits equal to (FRAMESZ + 1):

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• The minimum frame size is 8 bits. • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32 bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately. • If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO contains the remainder bits. For example, a 72-bit transfer consists of three words: the first and second words are 32 bits, and the third word is 8 bits.

66.4.5.1.17 Transmit Data (TDR)

Offset

Register	Offset
TDR	64h

Function

Pushes the data into the transmit FIFO, in the same order that the data is written. You can write to this register using 32-, 16-, or 8-bit writes.

When you write to this register or to Transmit Command (TCR), each write pushes data into the FIFO with zero pushed in unwritten bytes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DATA	Transmit Data Indicates transmit data. Both 8-bit and 16-bit writes of transmit data zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits) means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes.

66.4.5.1.18 Receive Status (RSR)

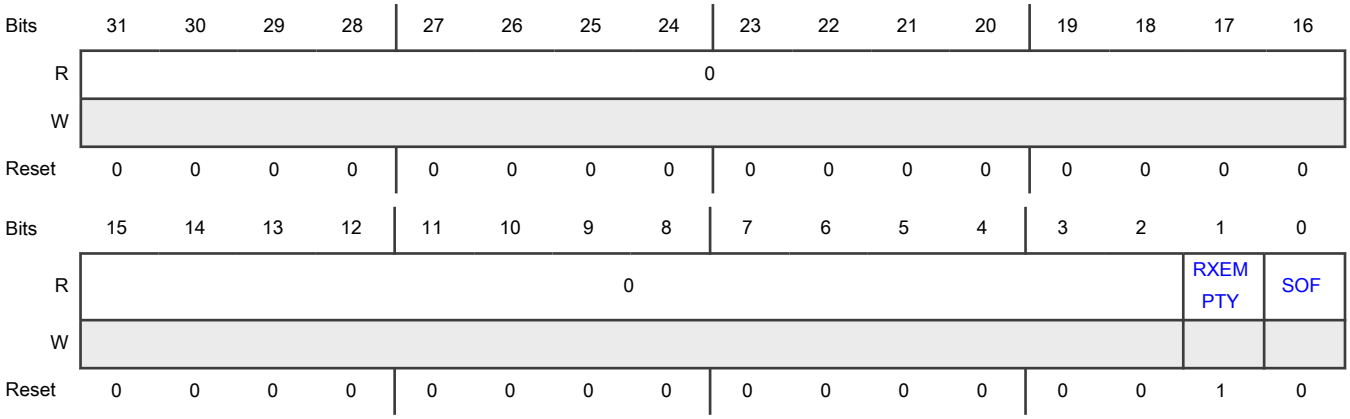
Offset

Register	Offset
RSR	70h

Function

Contains data flow status fields for receive FIFO.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 RXEMPTY	RX FIFO Empty Indicates whether the receive FIFO is empty. 0b - Not empty 1b - Empty
0 SOF	Start of Frame Indicates whether this is the first data word received after PCS assertion. 0b - Subsequent data word 1b - First data word

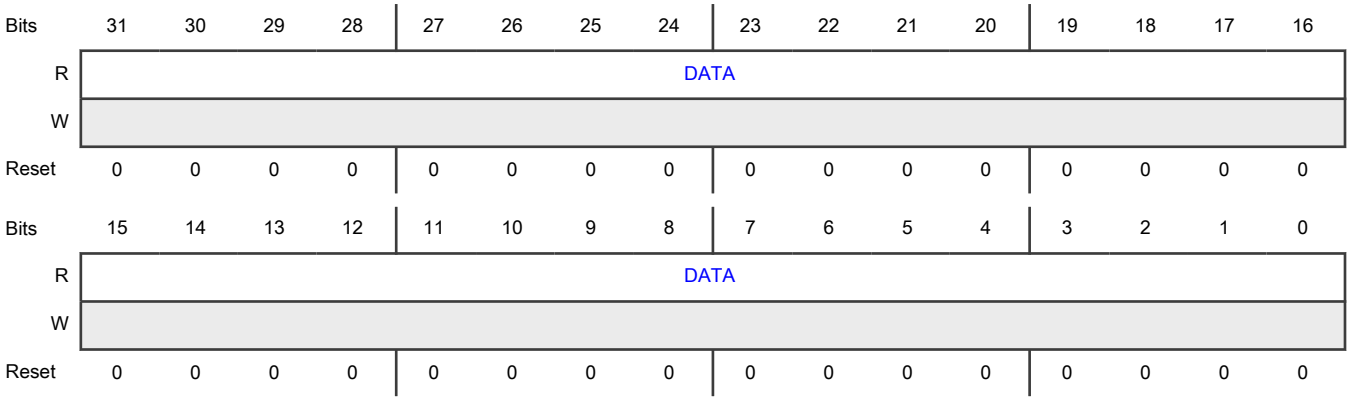
66.4.5.1.19 Receive Data (RDR)

Offset

Register	Offset
RDR	74h

Function
Pulls the first entry from the receive FIFO.

Diagram



Fields

Field	Function
31-0 DATA	Receive Data

66.4.5.1.20 Receive Data Read Only (RDROR)

Offset

Register	Offset
RDROR	78h

Function
Returns the first entry in the receive FIFO but does not remove the data from the FIFO.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DATA	Receive Data

66.4.5.1.21 Transmit Command Burst (TCBR)

Offset

Register	Offset
TCBR	3FCh

Function

Supports burst transfers of command data to the transmit FIFO for use with the DMA controller.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DATA	Command Data Writes data to Transmit Command (TCR).

66.4.5.1.22 Transmit Data Burst (TDBR0 - TDBR127)

Offset

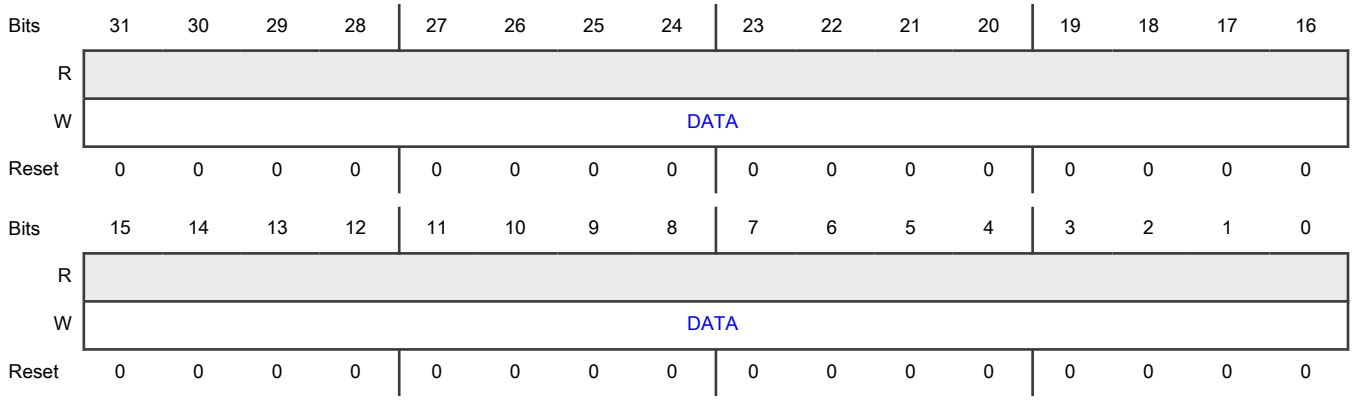
For n = 0 to 127:

Register	Offset
TDBRn	400h + (n × 4h)

Function

Supports burst transfers of data to the transmit FIFO for use with the DMA controller. The size of this register is 512 bytes.

Diagram



Fields

Field	Function
31-0 DATA	Data Writes data to Transmit Data (TDR).

66.4.5.1.23 Receive Data Burst (RDBR0 - RDBR127)

Offset

For n = 0 to 127:

Register	Offset
RDBRn	600h + (n × 4h)

Function
Supports burst transfers of data from the receive FIFO. The size of this register is 512 bytes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DATA	Data Reads data from Receive Data (RDR).

66.5 Low-Power Universal Asynchronous Receiver/Transmitter (LPUART)

66.5.1 Overview

LPUART provides asynchronous, serial communication capabilities with external devices. It supports the non-return-to-zero (NRZ) encoding format and infrared data association (IrDA)-compatible, low-speed serial infrared (SIR) protocol. LPUART can continue operating when the processor is in Low-Power mode, if an appropriate peripheral clock is available.

66.5.1.1 Block diagram

Figure 381 shows the transmitter portion of LPUART.

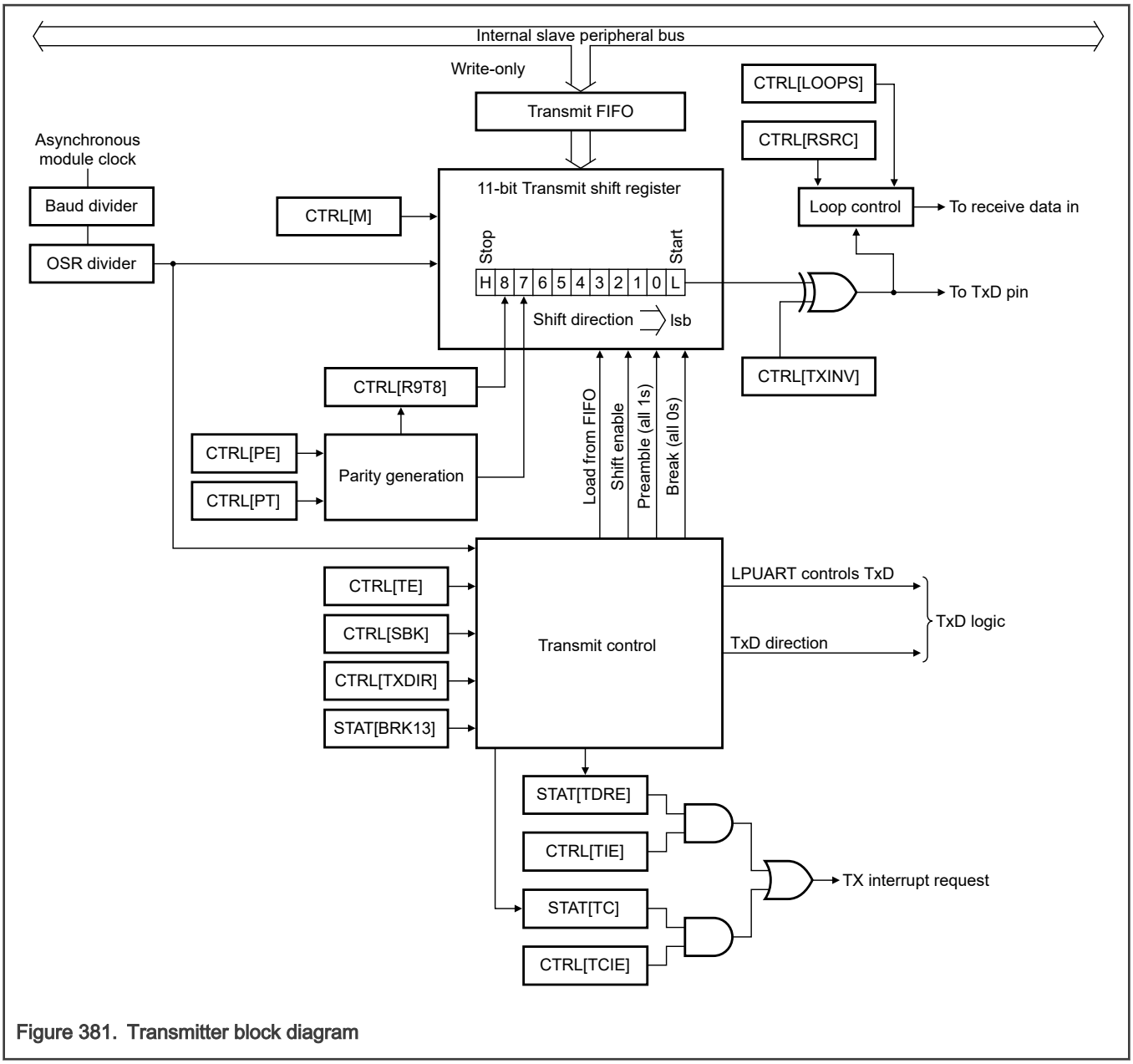


Figure 381. Transmitter block diagram

Figure 382 shows the receiver portion of LPUART.

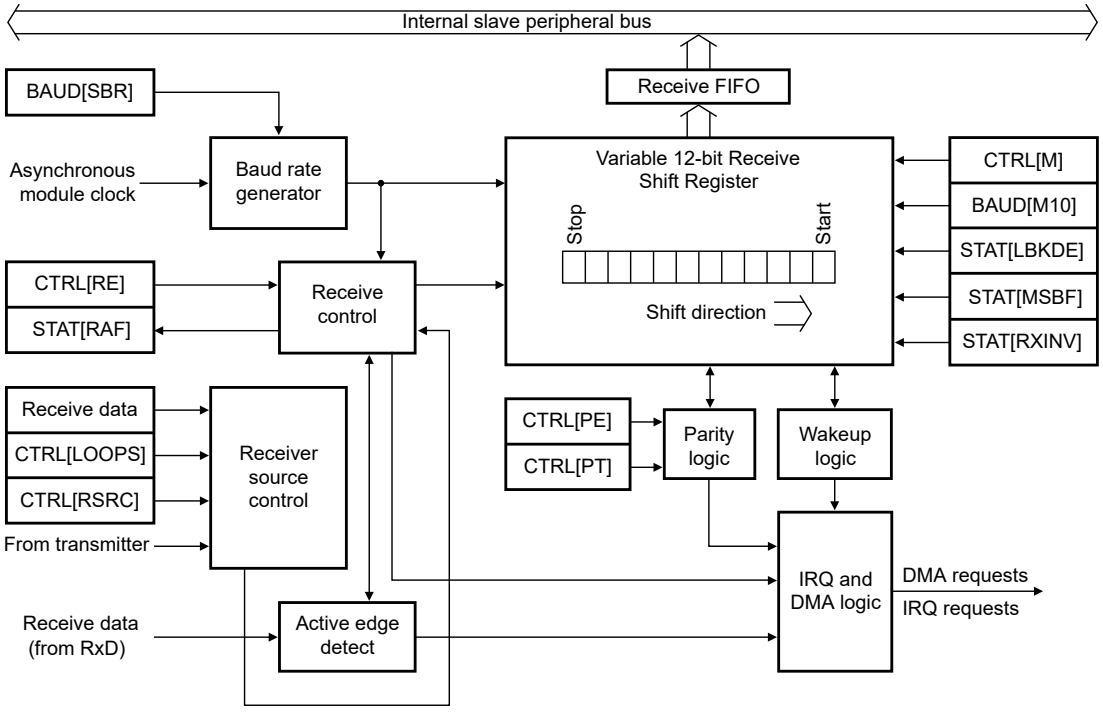


Figure 382. Receiver block diagram

66.5.1.2 Features

- Full-duplex, standard NRZ format
- Programmable baud rates (13-bit modulo divider) with a configurable oversampling ratio (OSR) from 4× to 32×
- Asynchronous operation of transmit and receive baud rates with respect to the bus clock:
  - Baud rate can be configured independently of the bus clock frequency.
  - Operation in Low-Power modes is supported.
- Interrupt, DMA, or polled operations:
  - Transmit data empty and transmission complete
  - Receive data full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
  - Receive data match
- Hardware parity generation and checking
- Programmable 7-bit, 8-bit, 9-bit, or 10-bit character length
- Programmable 1-bit or 2-bit stop bits
- Support for three receiver wake-up methods:
  - Idle line wake-up

  - Address mark wake-up
  - Receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address match start, address match end
- Optional 13-bit and 11-bit break character generation
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64, or 128 idle characters
- Selectable transmitter output and receiver input polarity
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with a programmable pulse width
- Independent FIFO structure for transmit and receive functions:
  - Separate configurable watermarks for receive and transmit requests
  - Option for receiver to assert request after a configurable number of idle characters, if receive FIFO is not empty

66.5.2 Functional description

LPUART supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. The following sections describe all LPUART blocks.

66.5.2.1 Baud rate generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and transmitter. The value, ranging from 1 to 8191, written to BAUD[SBR] determines the baud clock divisor for the asynchronous LPUART baud clock. The baud rate clock drives the receiver, while a bit clock, generated from the baud rate clock divided by the OSR, drives the transmitter. Depending on the OSR, the receiver has an acquisition rate of 4 to 32 samples per bit time.

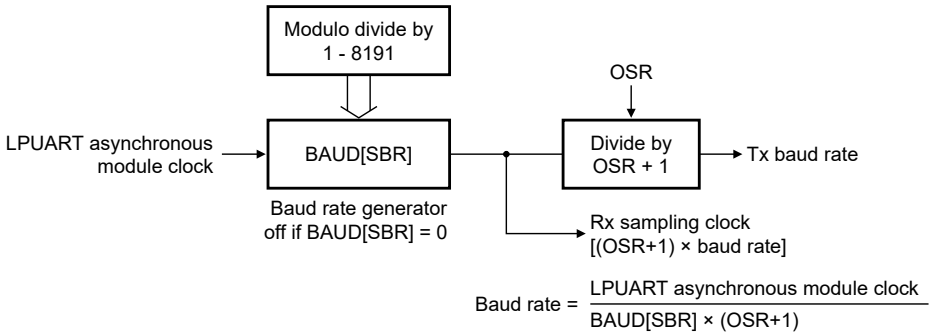


Figure 383. Baud rate generation

Baud rate generation is subject to these sources of error:

- Integer division of the asynchronous LPUART baud clock may not give the exact target frequency.
- Synchronization with the asynchronous LPUART baud clock can lead to a phase shift.

Baud rate generation is a free-running counter that continues whenever the transmitter or receiver is enabled. The transmitter bit clock continues whenever the transmitter is enabled; each transmitted character aligns to the next edge of the transmit bit clock.

In general, configuring OSR for a higher ratio and/or sampling on both edges of the clock slightly improves LPUART's tolerance to baud rate mismatch between the received data and LPUART configured baud rate. However, the three data samples in each bit (see Data sampling technique) are also closer together, which may impact noise sensitivity.

66.5.2.2 Baud rate tolerance

A transmitting device may operate at a baud rate below or above that of the receiver.

Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. A noise error will occur if the three samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the three stop bit samples are a logic zero.

As the receiver samples an incoming frame, it may re-synchronize the oversampling clock on any valid falling edge within the frame. Resynchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

In general, increasing the number of samples per bit will increase the baud rate tolerance and decreasing the number of samples per bit will reduce the baud rate tolerance. Note that since LPUART implements triple voting on consecutive receive data samples, increasing the number of samples per bit will move those samples closer together which would reduce the width of noise that can be filtered by the triple voting logic.

66.5.2.3 Calculating baud rate tolerance

Using the following definitions:

- SAM is the number of sample points per bit (valid range from 8 to 32; equal to (OSR + 1) × (BOTHEDGE + 1)).
- BIT is the number of bits in a character including start, data and stop bits (valid range from 9 to 13).

The ideal baud rate tolerance can be calculated as follows:

- Slow data rate tolerance = ((SAM ÷ 2) - 1) ÷ ((SAM × BIT) - (SAM ÷ 2) + 2)
- Fast data rate tolerance = ((SAM ÷ 2) - 2) ÷ (SAM × BIT)

As an example, if configured for 8-bit data, 1 stop bit (BIT = 10) and with OSR=0x7 and BOTHEDGE = 1 (SAM = 16):

- Slow data rate tolerance = (8 - 1) ÷ (160 - 8 + 2) = 7 ÷ 154 = 4.54%
- Fast data rate tolerance = (8 - 2) ÷ 160 = 6 ÷ 160 = 3.75%

If configured for 9-bit data with 1 stop bit (BIT = 11) with same oversampling configuration, then:

- Slow data rate tolerance = (8 - 1) ÷ (176 - 8 + 2) = 7 ÷ 170 = 4.12%
- Fast data rate tolerance = (8 - 2) ÷ 176 = 6 ÷ 176 = 3.41%

NOTE

Additional factors can contribute to a lower baud rate tolerance than the ideal. These include clock uncertainty or jitter on the LPUART functional clock source, differences in rise and fall times on the transmitter output and synchronization of the external receive pin to the local LPUART functional clock.

66.5.2.4 Transmitter functional description

This section describes the functioning of the LPUART transmitter, as shown in the transmitter portion of Block diagram, as well as specialized functions for sending break and idle characters.

The transmitter output (TXD) idle state defaults to logic high; the transmitter output is inverted when you write 1 to CTRL[TXINV], which becomes 0 following reset. You can enable the transmitter by writing 1 to CTRL[TE]. This queues a preamble character that is one full character frame of the Idle state. The transmitter then remains idle until data is available in the transmit FIFO and programs store data in the transmit FIFO by writing to Data (DATA).

The central element of the LPUART transmitter is the transmit shift register that is 9-bit to 13-bit long depending on the settings of CTRL[M], CTRL[M7], BAUD[M10], and BAUD[SBNS]. Going forward in this discussion, assume that CTRL[M], CTRL[M7], BAUD[M10], and BAUD[SBNS] are 0, selecting the normal 8-bit Data mode, in which the shift register holds a start bit, eight data

bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in transmit FIFO is transferred to the transmit shift register, synchronized with the baud rate clock, and STAT[TDRE] becomes 1 to indicate that another character may be written to the transmit FIFO at Data (DATA).

If no new character is waiting in the transmit FIFO after a stop bit is shifted out of the TXD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TXD high, waiting for more characters to transmit.

Writing 0 to CTRL[TE] does not immediately disable the transmitter. The current transmit activity in progress must first be completed (that could include a data character, idle character, or break character), although the transmitter does not start transmitting another character.

66.5.2.4.1 Break character length

CTRL[SBK] sends break characters, originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0, 9-bit to 12-bit times, including the start and stop bits. You can enable a longer break of 13-bit times by writing 1 to STAT[BRK13]. Normally, a program waits for STAT[TDRE] to become 1 to indicate that the last character of a message has moved to the transmit shifter. Next, the program writes 1 and then writes 0 to CTRL[SBK]. This action queues a break character to be sent as soon as the shifter is available. If CTRL[SBK] remains 1 when the queued break moves into the shifter, synchronized with the baud rate clock, an additional break character is queued. When LPUART is the receiving module, it receives a break character as 0s in all data bits and a framing error (STAT[FE] = 1) is detected.

You can also transmit a break character by writing to Data (DATA) with DATA[FRETSC] = 1 and the data bits clear. This supports transmitting the break character as part of the normal data stream and also allows DMA to transmit a break character.

When idle line wake-up is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program waits for STAT[TDRE] to become 1 to indicate that the last character of a message has moved to the transmit shifter. Next, write 0 and then write 1 to CTRL[TE]. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while CTRL[TE] becomes 0, the LPUART transmitter does not release control of the TXD pin.

You can also write to Data (DATA) to transmit an idle character, with DATA[FRETSC] and DATA[R9T9] = 1 and the values of all the other fields = 0. This supports transmitting the idle character as part of the normal data stream and also allows DMA to transmit an idle character.

As shown in the following table, STAT[BRK13], CTRL[M], CTRL[M7], BAUD[M10], and BAUD[SBNS] affect the length of the break character.

Table 574. Break character length

STAT[BRK13]	CTRL[M]	BAUD[M10]	CTRL[M7]	BAUD[SBNS]	Break character length (in bit times)
0	0	0	0	0	10
0	0	0	0	1	11
0	0	0	1	0	9
0	0	0	1	1	10
0	1	0	—	0	11
0	1	0	—	1	12
0	—	1	—	0	12
0	—	1	—	1	13
1	0	0	0	0	13
1	0	0	0	1	13
1	0	0	1	0	12

Table continues on the next page...

Table 574. Break character length (continued)

STAT[BRK13]	CTRL[M]	BAUD[M10]	CTRL[M7]	BAUD[SBNS]	Break character length (in bit times)
1	0	0	1	1	12
1	1	0	—	0	14
1	1	0	—	1	14
1	—	1	—	0	15
1	—	1	—	1	15

66.5.2.4.2 Hardware flow control

The transmitter supports hardware flow control by gating the transmission with the value of CTS_B. If the CTS operation is enabled, the character is transmitted when CTS_B is asserted. If CTS_B is deasserted in the middle of a transmission with characters remaining in the transmitter FIFO, the character in the transmit shift register is complete. Any characters in the FIFO wait for CTS_B to assert again, and TXD remains in the mark state (idle state) until CTS_B is reasserted. The CTS_B pin must assert for longer than one bit period to guarantee that a new transmission is started when the transmitter is idle with CTS.

If the CTS operation is disabled, the transmitter ignores the state of CTS_B.

The transmitter's CTS_B signal can be enabled even if the same LPUART receiver's RTS_B signal is disabled.

66.5.2.4.3 Transceiver driver enable

The transmitter can use RTS_B as an enable signal for the driver of an external transceiver. See Transceiver driver enable using RTS_B for details. If the RTS operation is enabled, when a character is placed into an empty transmit shift register, RTS_B asserts 1-bit time before the start bit is transmitted. RTS_B remains asserted for the whole time that the transmit shift register has any characters. RTS_B deasserts 1-bit time after all characters in the transmit FIFO and shift register are completely sent, including the last stop bit. In other words, when RTS_B is used as a transceiver enable, RTS_B asserts 1-bit time before the transmitter starts transmitting and negates 1-bit time after the transmitter goes idle.

Transmitting a break character also asserts RTS_B, with the same assertion and deassertion timing as having a character in the transmit shift register.

The transmitter's RTS_B signal asserts only when the transmitter is enabled. However, the transmitter's RTS_B signal is unaffected by its CTS_B signal. RTS_B remains asserted until the transfer is complete, even if the transmitter is disabled mid-way through a data transfer.

You can configure HDCR[RTSEXT] to the desired length by delaying the transmitter's RTS_B negation by up to 256-bit clock (baud rate) after the last stop bit.

66.5.2.4.4 Transceiver driver enable using RTS_B

RS-485 is a multiple drop communication protocol in which the LPUART transceiver's driver is three-stated unless LPUART is driving. The transmitter can use the RTS_B signal to enable the driver of a transceiver. The polarity of RTS_B can be matched to the polarity of the transceiver's driver enable signal.

The following figure shows the receiver enable signal asserted. This connection can also connect RTS_B to both DE and RE_B. The transceiver's receiver is disabled when driving. A pullup can pull RXD to a nonfloating value during this time. You can refine this option further by operating LPUART in Single-Wire mode, freeing the RXD pin for other uses.

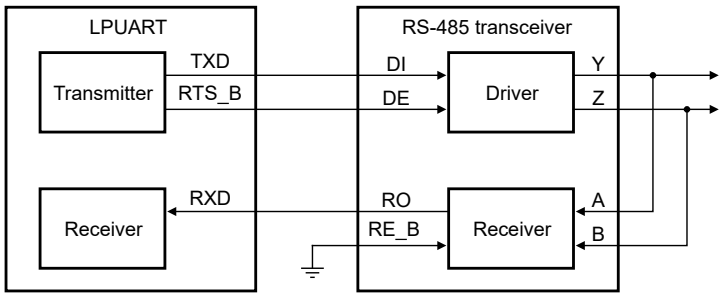


Figure 384. Transceiver driver enable using RTS_B

66.5.2.5 Receiver functional description

This section discusses the functioning of the LPUART receiver, as shown in the receiver portion of Block diagram. The section also discusses:

- The data sampling technique used to reconstruct receiver data.
- Different variations of the receiver wake-up function.

You can invert the receiver input by writing 1 to STAT[RXINV] and enable the receiver by writing 1 to CTRL[RE]. Character frames consist of a start bit of logic 0, along with N (7, 8, 9, 10) bits (MSB or LSB first), and one or two stop bits of logic 1. For information about 7-bit, 9-bit, or 10-bit Data mode, see Data modes. Going forward in this discussion, assume that LPUART is configured for a normal 8-bit Data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full (STAT[RDRF] = 0), the data character is transferred to the receive FIFO, resulting in STAT[RDRF] becoming 1. However, if STAT[RDRF] is already 1, indicating that the receive data buffer is already full, STAT[OR] becomes 1 and the new data is lost.

Because the LPUART receiver is separate from the receive FIFO, the receive shift register can receive the next word when the receive FIFO is full, and it is only at the end of the character that the next data is written into the receive FIFO, potentially triggering the overrun flag if the FIFO is full.

When a program detects that the receive data register is full (STAT[RDRF] = 1), it gets the data from the FIFO by reading Data (DATA). See Interrupts for details about flag clearing.

66.5.2.5.1 Data sampling technique

The LPUART receiver supports a configurable oversampling rate of between 4× and 32× of the baud rate clock for sampling. The receiver starts by considering logic level samples at the oversampling rate times the baud rate to search for a falling edge on the RXD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The oversampling baud rate clock divides the bit time into 4 to 32 segments from 1 to OSR (where OSR is the configured oversampling ratio). When a falling edge is located, three more samples are taken at (OSR ÷ 2), (OSR ÷ 2) + 1, and (OSR ÷ 2) + 2 to ensure that this is a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes they are synchronized to a received character. If another falling edge is detected before the receiver is considered synchronized, the receiver restarts sampling from the first segment.

The receiver then samples each bit time, including the start and stop bits, at (OSR ÷ 2), (OSR ÷ 2) + 1, and (OSR ÷ 2) + 2, to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, noise flag (STAT[NF]) becomes 1 when the received character is transferred to the receive FIFO.

When the LPUART receiver is configured to sample on both edges of the baud rate clock (that is, when BAUD[BOTHEDGE] = 1), the number of segments in each received bit is effectively doubled (from 1 to OSR× 2). The start and data bits are then sampled at OSR, OSR + 1, and OSR + 2. You must enable sampling on both edges of the clock for oversampling rates of 4× to 7×. This sampling is optional for higher oversampling rates.

The synchronization feature of LPUART synchronizes the internal oversampling counter with a detected falling edge on the receive signal, and to adjust the data sampling window. The falling edge detection needs three consecutive 1s prior to the "1->0" (one to zero) transition. After the initial falling edge detection for the start bit, the circuit continuously monitors the next falling edge, and resets the counter after another falling edge is detected. This synchronization to the start bit is termed as resynchronizarion.

When BAUD[RESYNCDIS] is 0, you perform this falling edge detection and resynchronization not only for the start bit but also for the rest of the character reception after the start bit.

When BAUD[RESYNCDIS] is 1, you perform the falling edge detection and resynchronization only for the start bit. The use case for disabling the resynchronization is protocols that require this (for example, LIN 2.1 prohibits resynchronization within a byte).

The following table and figure explain LPUART resynchronization.

Table 575. LPUART resynchronization settings

Resynchronization	BAUD[RESYNCDIS] = 0	BAUD[RESYNCDIS] = 1
For the starting bit falling edge	Yes	Yes
For all falling edges after the start bit	Yes	No

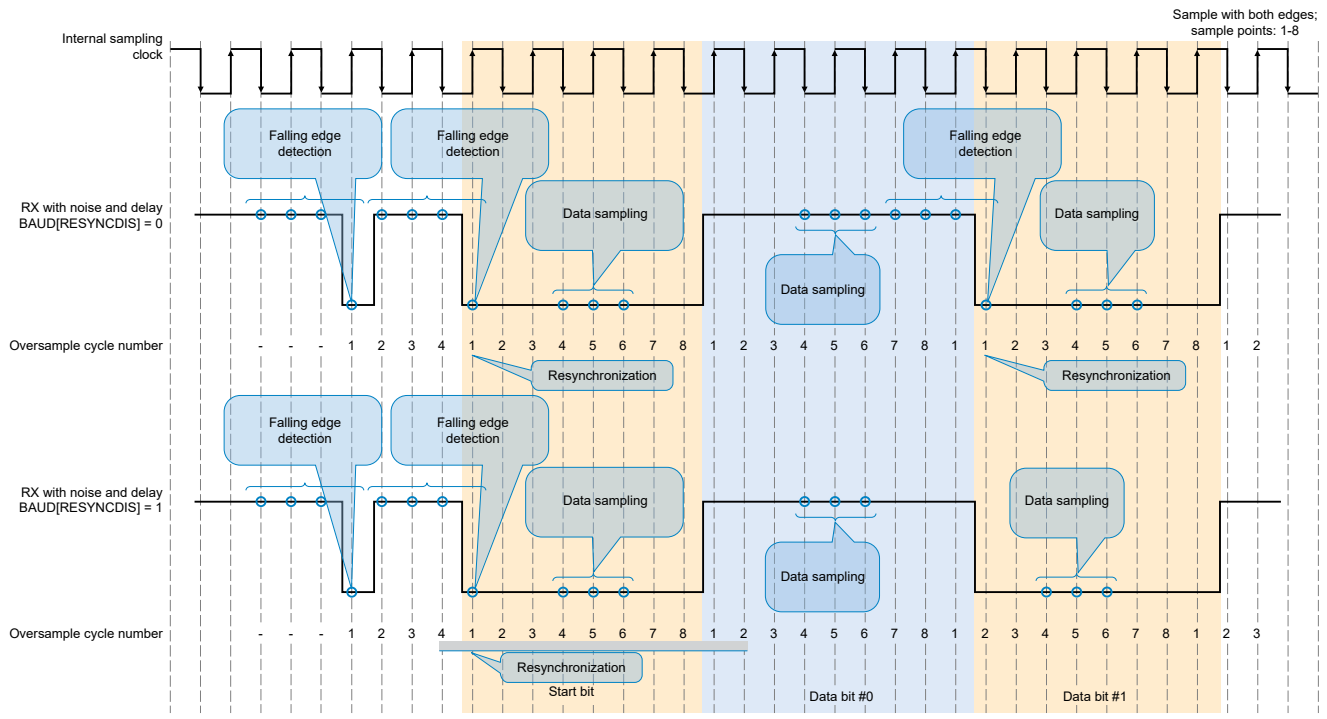


Figure 385. LPUART resynchronization diagram

66.5.2.5.2 Receiver wake-up operation

Receiver wake-up and receiver address matching are hardware mechanisms that allow an LPUART receiver to ignore the characters in a message intended for a different receiver.

During receiver wake-up, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write 1 to CTRL[RWU].

When CTRL[RWU] and STAT[RWUID] are 1, the status fields associated with the receiver, with the exception of STAT[IDLE], are inhibited from becoming 1, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, all receivers automatically force CTRL[RWU] to become 0. This results in all receivers waking up in time to look at the first character(s) of the next message.

During receiver address matching, the address matching is performed in hardware and the LPUART receiver ignores all characters that do not meet the address match requirements.

Table 576. Receiver wake-up options

CTRL[RWU]	BAUD[MAEN1] | BAUD[MAEN2]	BAUD[MATCFG]	CTRL[WAKE]: STAT[RWUID]	Receiver wake-up
0	0	X	X	Normal operation
1	0	00	00	Receiver wake-up on idle line; STAT[IDLE] = 0
1	0	00	01	Receiver wake-up on idle line; STAT[IDLE] = 1
1	0	00	10	Receiver wake-up on address mark
1	1	11	10	Receiver wake-up on data match
0	1	00	X0	Address mark address match; STAT[IDLE] = 0 for discarded characters
0	1	00	X1	Address mark address match; STAT[IDLE] = 1 for discarded characters
0	1	01	X0	Idle line address match
0	1	10	X0	Match on and match off; STAT[IDLE] = 0 for discarded characters
0	1	10	X1	Match on and match off; STAT[IDLE] = 1 for discarded characters

66.5.2.5.2.1 Idle line wake-up

When CTRL[WAKE] is 0, you can configure the receiver for an idle line wake-up. In this mode, CTRL[RWU] becomes 0 automatically when the receiver detects a full character time of the idle-line level.

CTRL[M], CTRL[M7], and BAUD[M10] select 7-bit to 10-bit Data mode and BAUD[SBNS] selects a 1-bit or 2-bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 9 to 13 bit times because of the start and stop bits.

When CTRL[RWU] is 1 and STAT[RWUID] is 0, the idle condition that wakes up the receiver does not lead to STAT[IDLE] becoming 1. The receiver wakes up and waits for the first data character of the next message that leads to STAT[RDRF] becoming 1 and generates an interrupt if enabled. When STAT[RWUID] is 1, any idle condition leads to STAT[IDLE] becoming 1 and generates an interrupt if enabled, regardless of whether CTRL[RWU] is 0 or 1.

These are the ways to detect an idle line:

- When CTRL[ILT] is 0, the idle bit counter starts after the start bit so that the stop bit and any logic 1s at the end of a character count to calculate the full character time of idle.
- When CTRL[ILT] is 1, the idle bit counter does not start until after the stop bit time so that the data in the last character of the previous message does not impact the idle detection.

66.5.2.5.2.2 Address mark wake-up

When CTRL[WAKE] is 1, you can configure the receiver for an address mark wake-up. In this mode, CTRL[RWU] becomes 0 automatically when the receiver detects a logic 1 in the most significant bit of the received character. When parity is enabled, the second most significant bit is used for address mark wake-up.

Address mark wake-up allows messages to contain idle characters, but requires one bit to be reserved for use in address frames. The logic 1 in the most significant bit (or second most significant bit when parity is enabled) of an address frame writes 0 to CTRL[RWU] and writes 1 to STAT[RDRF]. In this case, the character with the address mark bit is received even if the receiver is sleeping during most of this character time.

66.5.2.5.2.3 Data match wake-up

When CTRL[RWU] and CTRL[WAKE] are 1, and BAUD[MATCFG] equals 11, the receiver is configured for a data match wake-up. In this mode, CTRL[RWU] becomes 0 automatically when the receiver detects a character that matches MATCH[MA1] when BAUD[MAEN1] is 1, or that matches MATCH[MA2] when BAUD[MAEN2] is 1.

66.5.2.5.2.4 Address match operation

You can enable the address match operation when either BAUD[MAEN1] or BAUD[MAEN2] is 1 and BAUD[MATCFG] is 0. In this function, a character that the RXD pin receives with a logic 1 in the most significant bit (or the second most significant bit when parity is enabled) is considered an address and is compared to the associated MATCH[MA1] or MATCH[MA2]. The character is only transferred to the receive buffer, and STAT[RDRF] becomes 1 if the comparison matches. All subsequent characters received with a logic 0 in the most significant bit (or the second most significant bit when parity is enabled) are considered to be data associated with the address and are transferred to the receive FIFO. If no marked address match occurs, no transfer is made to the receive FIFO, and all the characters that follow, with logic 0 in the most significant bit (or second most significant bit when parity is enabled), are also discarded. If both BAUD[MAEN1] and BAUD[MAEN2] are 0, the receiver operates normally, and all the received data is transferred to the receive FIFO.

The address match operation functions in the same way for both MATCH[MA1] and MATCH[MA2]:

- If either BAUD[MAEN1] or BAUD[MAEN2] is 1, a marked address is compared only to the associated Match Address (MATCH) and data is transferred to the receive FIFO only on a match.
- If both BAUD[MAEN1] and BAUD[MAEN2] are 1, a marked address is compared to both MATCH[MA1] and MATCH[MA2] and data is transferred only on a match with either of these fields.

66.5.2.5.2.5 Idle match operation

You can enable the idle match operation when either BAUD[MAEN1] or BAUD[MAEN2] is 1 and BAUD[MATCFG] is 1. In this function, the first character that the RXD pin receives after an idle line condition is considered an address and is compared to the associated MATCH[MA1] or MATCH[MA2]. The character is transferred only to the receive buffer, and STAT[RDRF] becomes 1, if the comparison matches. All subsequent characters are considered to be data associated with the address and are transferred to the receive FIFO until the next idle line condition is detected. If no address match occurs, no transfer is made to the receive FIFO, and all the frames that follow, until the next idle condition, are also discarded. If both BAUD[MAEN1] and BAUD[MAEN2] are 0, the receiver operates normally, and all the received data is transferred to the receive FIFO.

An idle match operation functions in the same way for both MATCH[MA1] and MATCH[MA2]:

- If either BAUD[MAEN1] or BAUD[MAEN2] is 1, the first character after an idle line is compared only to the associated Data (DATA) and data is transferred to the receive FIFO only on a match.

- If both BAUD[MAEN1] and BAUD[MAEN2] are 1, the first character after an idle line is compared to both MATCH[MA1] and MATCH[MA2] and data is transferred only on a match with either of these fields.

66.5.2.5.2.6 Match on, match off operation

The match on, match off operation is enabled when both BAUD[MAEN1] and BAUD[MAEN2] are 1 and BAUD[MATCFG] = 10. In this function, a character that the RXD pin receives matches MATCH[MA1] and is transferred to the receive buffer, and STAT[RDRF] becomes 1. All subsequent characters are considered to be data and are also transferred to the receive FIFO, until a character that matches MATCH[MA2] is received. The character that matches MATCH[MA2], along with all subsequent characters, is discarded; and this continues until another character that matches MATCH[MA1] is received. If both BAUD[MAEN1] and BAUD[MAEN2] are 0, the receiver operates normally, and all the received data is transferred to the receive FIFO.

NOTE

The match on, match off operation requires both BAUD[MAEN1] and BAUD[MAEN2] to be 1.

66.5.2.5.3 Hardware flow control

To support hardware flow control, you can program the receiver to automatically assert and deassert RTS_B:

- RTS_B remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using RTS_B for more information.
- If the receiver RTS functionality is enabled, the receiver automatically deasserts RTS_B if STAT[RDRF] is 1 or a start bit is detected that causes STAT[RDRF] to become 1.
- The receiver asserts RTS_B when STAT[RDRF] is 0 and has not detected a start bit that causes STAT[RDRF] to become 1. There is no impact if STAT[RDRF] is 1 already.
- Even if RTS_B is deasserted, the receiver continues to receive characters until the receive FIFO is overrun.
- If the receiver RTS functionality is disabled, the receiver's RTS_B remains deasserted.

66.5.2.6 Additional LPUART functions

66.5.2.6.1 Data modes

You can configure the LPUART transmitter and receiver to operate in 7-bit Data mode by writing 1 to CTRL[M7], 9-bit Data mode by writing 1 to CTRL[M], or 10-bit Data mode by writing 1 to BAUD[M10]. In 9-bit Data mode, there exists a ninth data bit and in 10-bit mode, there exists a tenth data bit.

When performing 8-bit writes to the transmit FIFO, the ninth and tenth bits are pushed into the FIFO from CTRL[T8] and CTRL[T9]. For coherent 8-bit writes, you must write to CTRL[T8] and CTRL[T9] before writing to Data (DATA)[7:0]. However, if the values in CTRL[T8] or CTRL[T9] do not need to change, it is not necessary to update CTRL[T8] and CTRL[T9] before every 8-bit write to Data (DATA).

When performing 16-bit or 32-bit writes to the transmit FIFO, all 10 bits are pushed into the transmit FIFO from the write data.

When performing 8-bit reads of the receive FIFO, the ninth and tenth bits are held in CTRL[R8] and CTRL[R9] but you must read them before reading Data (DATA). A 16-bit or 32-bit read of the receive FIFO returns all 10 bits in Data (DATA).

The 9-bit Data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with the address mark wake-up so that the ninth data bit can serve as the wake-up bit. The 10-bit Data mode is typically used with parity and address mark wake-up so that the ninth data bit can serve as the wake-up bit and the tenth bit can serve as the parity bit. In custom protocols, the ninth and/or tenth bits can also serve as software-controlled markers.

66.5.2.6.2 Idle length

An idle character is one where the start bit, all data bits, and stop bits are in the mark position (idle state, generally logic 1). You can configure CTRL[ILT] to start detecting an idle character from the previous start bit (any data bits and stop bits count for idle character detection) or from the previous stop bit.

You can also use CTRL[IDLECFG] to configure the number of idle characters that must be received before an idle line condition is detected. This field configures the number of idle characters that must be received before STAT[IDLE] becomes 1, STAT[RAF] becomes 0, and DATA[IDLINE] becomes 1 with the next received character.

CTRL[IDLECFG] also affects the idle line wake-up and idle match operations. When either the address match or match on/off operation is enabled, writing 1 to STAT[RWUID] causes any discarded characters to be treated as idle characters.

After the extended idle time is enabled for the receiver, you can configure an idle line condition by using REIR[IDTIME], which specifies the number of bits (baud rate) since the last stop bit that is required for an idle condition to be detected. This replaces the configuration of CTRL[ILT] and CTRL[IDLECFG].

The transmitter can also enable the extended idle time. In this case, any idle character queued through the transmit FIFO forces the transmitter to be idled for the configured number of bit clocks before the idle character is read from the FIFO and the transmitter continues.

After you enable the transmitter extended idle time, the transmitter does not automatically queue an idle character whenever it is enabled.

66.5.2.6.3 Loop mode

Enable Loop mode by setting CTRL[LOOPS] = 1 and CTRL[RSRC] = 0. You, sometimes, use Loop mode to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and LPUART does not use the RXD pin.

Loop mode also internally connects the RTS_B output to the CTS_B input and the DTR_B output to the DSR_B input.

66.5.2.6.4 Single-Wire mode

Enable Single-Wire mode by setting CTRL[LOOPS] = 1 and CTRL[RSRC] = 1. Single-Wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and TXD pin (the RXD pin is not used).

In Single-Wire mode, CTRL[TXDIR] controls the direction of serial data on the TXD pin. When CTRL[TXDIR] becomes 0, the TXD pin is an input to the receiver and the transmitter is temporarily disconnected from the TXD pin so that an external device can send serial data to the receiver. When CTRL[TXDIR] = 1, the TXD pin is an output that the transmitter drives. The internal loop back connection is disabled, and as a result, the receiver is unable to receive characters that the transmitter sends out.

Half Duplex Control (HDCR) replaces the implementation of CTRL[LOOPS] and CTRL[RSRC], and you can use this register to configure various options for both Single-Wire and Half-Duplex operations, using independent RXD and TXD pins:

- HDCR[TXSTALL] replaces the CTRL[TXDIR] functionality and prevents the transmitter from becoming busy or asserting the RTS_B transmitter if STAT[RAF] is 1.
- You can select the TXD pin, as the source for the receiver, to configure HDCR[RXSEL] for a single-wire operation. If HDCR[RXSEL] is 1, you must configure the TXD pin for an open-drain operation.
- HDCR[RXMSK] masks the receiver input when the RTS_B transmitter is asserted (this applies even if you have not configured RTS_B as an output).
- HDCR[RXWRMSK] blocks storage of the receive data in the receive FIFO when the RTS_B transmitter is asserted. This setting does not affect the receiver idle functionality.
- HDCR[RTSEXT] delays the negation of the RTS_B transmitter by the configured number of bit clocks (baud rate).

66.5.2.6.5 Timeout counter

LPUART implements four general-purpose timeout counters; counters 0 and 1 are used to monitor the receiver and counters 2 and 3 are used to monitor the transmitter. When enabled, you can configure each counter to monitor one of the following conditions:

- Idle time in number of bits, starting to increment when first enabled and an idle condition is detected. The counter restarts whenever a character is received or transmitted.
- Idle time in number of bits, starting to increment after the next character is received or transmitted. The counter restarts whenever a character is received or transmitted.

- Idle time is more than the timeout interval, in number of bits, but less than the extended idle timeout. The counter restarts whenever a character is received or transmitted. You can use this to detect a gap between characters that is greater than a threshold (timeout) but less than the configured extended idle time. This configuration requires the extended idle feature for the transmitter or receiver to be enabled for a proper operation.
- Number of characters received or transmitted is equal to the configured timeout. The counter asserts at the start of the character that equals the timeout value.

The timeout counters restart counting whenever the counter is disabled and then enabled. These timeout counters are disabled when the corresponding STAT[TSF] field is 1. If the timeout enable is still set after STAT[TSF] becomes 0, the timeout counter restarts as if the counter had been disabled and then enabled. You can measure the idle time in bit times from the end of the last stop bit and until a start bit is validated.

66.5.2.7 Peripheral triggers

The connection of the LPUART peripheral triggers with other peripherals is chip-specific.

66.5.2.7.1 Output triggers

LPUART generates the following output triggers that can be connected to other peripherals on the chip:

- The transmit word trigger asserts at the end of each transmitted word and negates after 1-bit period.
- The transmit data trigger is identical to the TXD pin output, but without support for input trigger modulation.
- The receive word trigger asserts at the end of each received word that is written to the receive FIFO, for one oversampling clock period.
- The receive idle trigger asserts when STAT[IDLE] becomes 1, and negates when the next valid start bit is detected.

66.5.2.7.2 Input trigger

LPUART supports a peripheral input trigger that you can configure in one of the following ways:

- By enabling the CTS function: You can connect the input trigger instead of the CTS_B pin input. The input trigger must assert for longer than 1-bit clock period when the transmitter is idle, with data to send, to guarantee a new transmission.
- By making the input trigger modulate the transmit data output (trigger is logically ANDed with the TXD output): The input trigger is expected to be a free-running clock (carrier signal) that generates from a timer or PWM source with a frequency that is greater than the bit-clock frequency. The carrier signal must not toggle faster than the maximum supported bit time.
- By connecting the input trigger instead of the RXD pin input: The input trigger is expected to be generated from a receive data source, such as an analog comparator or external pin.

66.5.2.8 Infrared (IR) interface

LPUART provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses, transforming them to serial bits, which are then sent to LPUART. The IrDA physical layer specification defines a half-duplex IR communication link for exchanging data. The full standard includes data rates up to 16 Mbit/s. The LPUART IrDA support is limited to SIR mode that supports data rates only between 2.4 kbit/s and 115.2 kbit/s.

LPUART has an infrared transmit encoder and a receive decoder. The infrared decoder converts the received character from the IrDA format to the NRZ format, which the receiver uses. It also has an OSR oversampling baud rate clock counter that filters noise and indicates when a 1 is received. LPUART transmits serial bits of data, which the infrared submodule encodes, to transmit a narrow pulse for every zero bit. No pulse is transmitted for every single bit. When receiving data, an IR photo diode (external to LPUART) detects the IR pulses. The IR receive decoder transforms them to CMOS levels. The infrared receive decoder then stretches the narrow pulses to get back to a serial bit stream that LPUART receives. You can invert the polarity of transmitted pulses and expected receive pulses so that a direct connection can be made to external IrDA transceiver modules that use active-high pulses.

The IR submodule receives its clock sources from LPUART. The submodule selects one of these clocks to generate either 1 ÷ OSR, 2 ÷ OSR, 3 ÷ OSR, or 4 ÷ OSR narrow pulses during transmission.

66.5.2.8.1 Infrared transmit encoder

The infrared transmit encoder converts serial bits of data from the transmit shift register to the TXD signal. A narrow pulse is transmitted for a 0 bit and no pulse is transmitted for a 1 bit. The narrow pulse is sent at the start of the bit with a duration of 1 ÷ OSR, 2 ÷ OSR, 3 ÷ OSR, or 4 ÷ OSR of a bit time. A narrow low pulse is transmitted for a 0 bit when CTRL[TXINV] is 0, while a narrow high pulse is transmitted for a 0 bit when CTRL[TXINV] is 1.

66.5.2.8.2 Infrared receive decoder

The infrared receive block converts data from the RXD signal to the receive shift register. A narrow pulse is expected for each 0 received and no pulse is expected for each 1 received. A narrow low pulse is expected for a 0 bit when STAT[RXINV] is 0, while a narrow high pulse is expected for a 0 bit when STAT[RXINV] is 1. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

66.5.2.8.3 Start-bit detection

When STAT[RXINV] is 0, the first falling edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently of each other.

66.5.2.8.4 Noise filtering

The decoder ignores any rising edges detected during the first half of the infrared decoder counter, and can leave any pulses less than one oversampling baud clock as undetected. This is regardless of whether the pulse is seen in the first or second half of the count.

66.5.2.8.5 Low-bit detection

During the second half of the decoder count, a rising edge is decoded as 0, which is sent to the receiver. The decoder counter is also reset.

66.5.2.8.6 High-bit detection

At OSR oversampling baud rate clocks after the previous rising edge, if a rising edge is not seen, the decoder sends a 1 to the receiver.

If the next bit is 0, which arrives late, a low bit is detected according to Low-bit detection. The value sent to the receiver is changed from 1 to 0. Then, if a noise pulse occurs outside the receiver's bit time sampling period, the delay of a 0 is not recorded as noise.

66.5.2.9 MODBUS protocol

66.5.2.9.1 MODBUS frame structure

MODBUS is an application layer messaging protocol, positioned at level 7 of the OSI model. It provides client and server communication between devices connected on different types of buses or networks.

The MODBUS protocol defines a simple protocol data unit (PDU) independent of the underlying communication layers. The mapping of the MODBUS protocol on specific buses or network can introduce some additional fields on the application data unit (ADU).

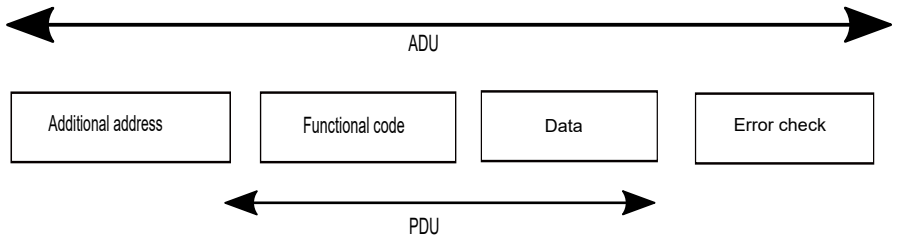


Figure 386. MODBUS frame structure

The function code field informs the server about which action to perform, as requested by the client (only the client can initiate the communication), where the data field contains additional information that the server uses to take the action defined by the function code. MODBUS PDU for serial line communication = 256 - server address (1 byte) - CRC (2 bytes) = 253 bytes.

You can set up a MODBUS controller to communicate on standard MODBUS networks using either of the two transmission modes: ASCII or RTU. RTU mode allows better character density and throughput for the same baud rate as the ASCII format allows 7 bits to represent a character where RTU keeps 8 bits for data representation in each packet.

66.5.2.9.2 MODBUS frame for LPUART

Though the protocol resides in the application layer, it can also be checked at the physical layer using LPUART by considering the frame structure shown in the following figure, following RTU mode of data transmission.

Start	Address	Function	Data	CRC	End
3.5 Char time	8 Bit	8 Bit	N * 8Bit	16 Bit	3.5 Char time

Figure 387. MODBUS frame for LPUART

An entire message frame of MODBUS must contain start character, address, function code, data, and end character. LPUART does not support CRC calculation for TX and RX, instead, each LPUART packet containing frame information is transmitted with its individual parity, and the same is checked in RX. Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message and a new message can begin after this interval. The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before the completion of the frame, the receiving device flushes the incomplete message, and assumes that the next byte is the address field of a new message. The correctness of timeout logic ensures the correctness of the MODBUS frame using LPUART.

Registers such as REIR, TEIR, TOCR, TOSR, TIMEOUT[0 - 3], TCBR, and TDBR are useful for realizing MODBUS using LPUART. The TX IDLE time is configured by using TEIR[IDTIME].

For higher baud rates, MODBUS recommends adopting fixed t1.5 and t3.5 times of 750us ms and 1.75 ms. The receiver idle line wake-up can be used by a MODBUS device to only wake-up on a matching address or broadcast address received after a valid idle time. The receiver end-of-packet DMA transfer can be used to offload the reception of a MODBUS packet onto the DMA controller.

66.5.2.9.3 MODBUS TX and RX programming sequence

The following programming sequences can be useful for sending and receiving MODBUS frames using LPUART.

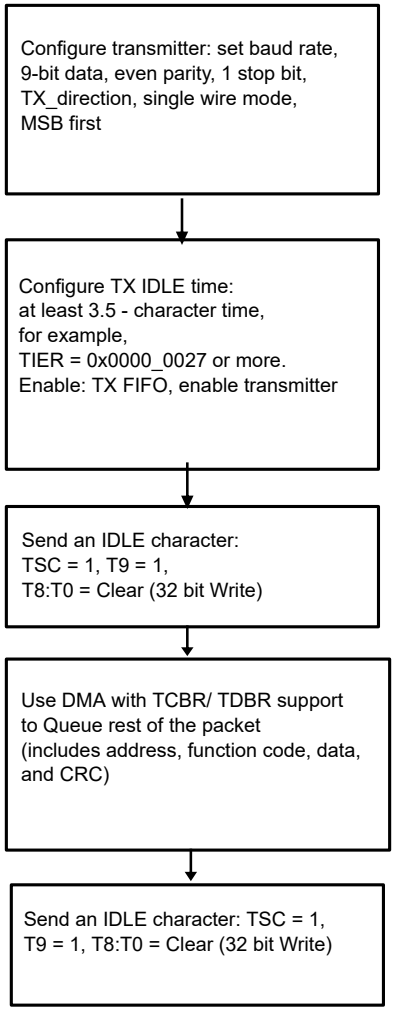


Figure 388. MODBUS TX programming

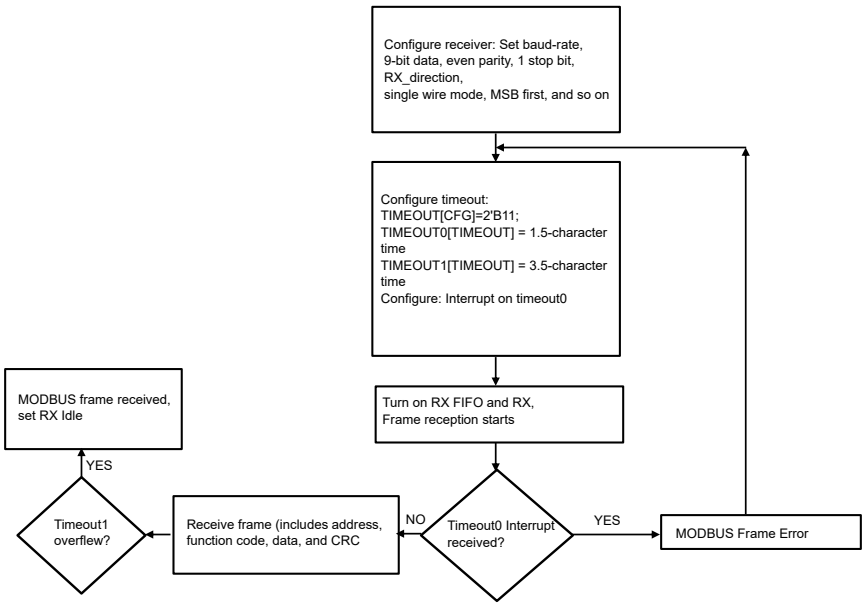


Figure 389. MODBUS RX programming

66.5.2.10 Modes of operation

66.5.2.10.1 Low-Power modes

NOTE

See the chip-specific information for specific low-power modes available on your chip.

66.5.2.10.2 Debug mode

LPUART remains functional in Debug mode.

66.5.2.11 Clocking

Table 577. Types of clocks

Clock	Description
Functional	Is asynchronous to the bus clock and can remain enabled in Low-Power modes to support transmit and/or receive functions, including low-power wake-up.
Bus	Is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPUART transmit and receive registers, including the FIFOs.

66.5.2.12 Reset

Table 578. Types of resets

Reset	Description
Chip	Enables the logic and registers for the LPUART transmitter and receiver to reset to their default states.

Table continues on the next page...

Table 578. Types of resets (continued)

Reset	Description
Software	Resets the LPUART logic and registers to their default states, except for Global (GLOBAL). GLOBAL[RST] controls the LPUART software reset.
FIFO	Implements write-only control fields that reset the transmit FIFO (FIFO[TXFLUSH]) and receive FIFO (FIFO[RXFLUSH]). After a FIFO is reset, that FIFO becomes empty.

66.5.2.13 Interrupts

The LPUART transmitter has two status fields that can optionally generate hardware interrupt requests. If STAT[TDRE] is 1, it indicates that there is room in the transmit FIFO to write another transmit character to Data (DATA). If CTRL[TIE] is 1, a hardware interrupt is requested when STAT[TDRE] is 1.

STAT[TC] indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TXD at the inactive level. This field is often used in systems with modems to determine when it is safe to turn off the modem. If CTRL[TCIE] is 1, a hardware interrupt is requested when STAT[TC] is 1. Instead of hardware interrupts, software polling may be used to monitor STAT[TDRE] and STAT[TC] if the corresponding CTRL[TIE] or CTRL[TCIE] field is 0.

When a program detects that STAT[RDRF] is 1, it gets the data from this field by reading Data (DATA). The field becomes 0 by reading Data (DATA).

STAT[IDLE] includes logic that prevents it from becoming 1 repeatedly when the RXD line remains idle for an extended period of time. STAT[IDLE] becomes 0 when you write 1 to it, and cannot become 1 again until the receiver has received at least one new character and has 1 as the value of STAT[RDRF].

If the associated error is detected in the received character that caused STAT[RDRF] to become 1, STAT[NF], STAT[FE], and STAT[PF] become 1 at the same time STAT[RDRF] becomes 1. These flags do not become 1 in overrun cases.

If STAT[RDRF] is already 1 when a new character is ready to be transferred from the receive shifter to the receive FIFO, STAT[OR] becomes 1, instead of the data along with any associated STAT[NF], STAT[FE], or STAT[PF] condition getting lost.

If the received character matches the contents of MATCH[MA1] and/or MATCH[MA2], then STAT[MA1F] and/or STAT[MA2F] become 1 at the same time that STAT[RDRF] becomes 1.

At any time, an active edge on the RXD serial data input pin causes STAT[RXEDGIF] to become 1. STAT[RXEDGIF] becomes 0 when you write 1 to it. This function depends on the receiver being enabled (the value of CTRL[RE] being 1).

MODEM Status (MSR) can generate an interrupt from a configured status field, which STAT[MSF] indicates.

Timeout Status (TOSR) can generate an interrupt from a configured status field, which STAT[TSF] indicates.

66.5.2.14 DMA

66.5.2.14.1 DMA burst support

To support efficient DMA transfers to the transmit FIFO, two alias regions are implemented to support incrementing 8-bit, 16-bit, or 32-bit write accesses to the transmit FIFO:

- Transmit Command Burst (TCBR0 - TCBR127) is a 512-byte region that supports pushing 16-bit data into the transmit FIFO.
- Transmit Data Burst (TDBR0 - TDBR255) is a 1024-byte region that supports pushing zero extended 8-bit data into the transmit FIFO.

The aforementioned regions are contiguous, so a DMA transfer can start in Transmit Command Burst (TCBR0 - TCBR127) to initialize the transfer including address mark, idle word, or break character, and then complete the transfer in Transmit Data Burst (TDBR0 - TDBR255) with the data to transmit, without changing the transfer size.

The transmit FIFO block writes overflow the FIFO, but that does not signal an error. Do not perform 32-bit writes to Transmit Data Burst (TDBR0 - TDBR255) unless there are four empty slots in the transmit FIFO, and do not perform 16-bit writes to this register unless there are two empty slots in the transmit FIFO.

66.5.2.14.2 End-of-packet DMA transfers

The end-of-packet functionality is designed for serial interfaces where you may not know the size of the transfer in advance and the data is being pushed by an external device. The end-of-packet processing ensures that data does not become stranded in either the receive FIFO or the DMA receive buffer. Support for end-of-packet processing must be implemented in both the serial interfaces and DMA controller.

The condition that signals the end of packet is different for each serial interface, but the serial peripheral and DMA process it in the same way. For example, an idle line condition signals the UART end of packet, the Stop and/or Repeated Start condition signals the I2C end of packet, and PCS negation signals the SPI end of packet.

When the serial peripheral is configured to signal the end-of-packet condition to the DMA and the serial interface detects an end-of-packet condition, it asserts the DMA request for the receive FIFO irrespective of the watermark configuration. For larger watermark configurations, this ensures that the last few words of the transfer are first flushed from the receive FIFO.

The DMA then reads the contents of the receive FIFO, depending on the first word in the FIFO:

- If the receive FIFO is empty, the serial interface signals an end of packet condition to the DMA controller.
- If the receive FIFO is not empty, but the first word in the FIFO is the start of a new packet, data is not pulled from the receive FIFO and the serial interface signals an end-of-packet condition to the DMA controller.
- If the receive FIFO is not empty, and the first word in the FIFO is not the start of a new packet, the DMA transfers the receive data as normal.

Because the DMA may be transferring multiple words on each request, the end-of-packet condition persists until the DMA minor loop has completed and no additional data is pulled from the receive FIFO. The field that triggered the end-of-packet condition becomes 0 when the minor loop completes following the end of packet being signaled to the DMA controller.

When the DMA detects the end-of-packet condition, it writes all received words up to the end of packet into the system memory and saves the destination address for the word after the last valid data. The DMA then terminates the channel as if the major loop completed, including final offsets and optional interrupts, channel linking, and scatter-gather. You can, optionally, save the final destination address to the system memory. The final destination address is also available in the destination address register.

Because the DMA terminates the major loop, no servicing of the receive FIFO occurs until you or the hardware reconfigures the DMA (for example, channel linking or scatter-gather). You must minimize this delay to avoid receiver FIFO overrun. The automatic DMA end-of-packet processing is not recommended when there are only a few words transferred between end-of-packet conditions. This is because the DMA spends more time processing the end of packet than transferring the data. For example, the UART idle line length must be increased as needed to avoid an excessive number of idle conditions.

66.5.3 External signals

Table 579. External signals

Signal	Description	I/O
TXD	Transmit data: This pin is normally an output, but is an input (tristated) in Single-Wire mode whenever the transmitter is disabled or the transmit direction is configured for receive data.	I/O
RXD	Receive data	I
CTS_B	Clear-to-send	I

Table continues on the next page...

Table 579. External signals (continued)

Signal	Description	I/O
RTS_B	Request-to-send	O
DTR_B	Data terminal ready	O
DSR_B	Data set ready	I
DCD_B	Data carrier detect	I
RIN_B	Ring indicator	I

66.5.4 Initialization

This module does not require initialization.

66.5.5 LPUART register descriptions

LPUART includes registers to control baud rate, select options, report status, and store transmit and receive data. Access to an address outside the valid memory map generates a bus error.

NOTE

Writing to a read-only (RO) register or reading a write-only (WO) register can cause bus errors. LPUART does not verify whether programmed values in the registers are correct; you must write valid values to them.

66.5.5.1 LPUART memory map

LPUART0 base address: 4009_2000h

LPUART1 base address: 4009_3000h

LPUART2 base address: 4009_4000h

LPUART3 base address: 4009_5000h

LPUART4 base address: 400B_4000h

LPUART5 base address: 400B_5000h

LPUART6 base address: 400B_6000h

LPUART7 base address: 400B_7000h

LPUART8 base address: 400B_8000h

LPUART9 base address: 400B_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0404_0007h
4h	Parameter (PARAM)	32	R	0000_0303h
8h	Global (GLOBAL)	32	RW	0000_0000h
Ch	Pin Configuration (PINCFG)	32	RW	0000_0000h
10h	Baud Rate (BAUD)	32	RW	0F00_0004h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
14h	Status (STAT)	32	RW	00C0_0000h
18h	Control (CTRL)	32	RW	0000_0000h
1Ch	Data (DATA)	32	RW	0000_1000h
20h	Match Address (MATCH)	32	RW	0000_0000h
24h	MODEM IrDA (MODIR)	32	RW	0000_0000h
28h	FIFO (FIFO)	32	RW	00C0_0022h
2Ch	Watermark (WATER)	32	RW	0000_0000h
30h	Data Read-Only (DATARO)	32	R	0000_1000h
40h	MODEM Control (MCR)	32	RW	0000_0000h
44h	MODEM Status (MSR)	32	RW	0000_0000h
48h	Receiver Extended Idle (REIR)	32	RW	0000_0000h
4Ch	Transmitter Extended Idle (TEIR)	32	RW	0000_0000h
50h	Half Duplex Control (HDCR)	32	RW	0000_0000h
58h	Timeout Control (TOCR)	32	RW	0000_0000h
5Ch	Timeout Status (TOSR)	32	RW	0000_000Fh
60h - 6Ch	Timeout N (TIMEOUT0 - TIMEOUT3)	32	RW	0000_0000h
200h - 3FCh	Transmit Command Burst (TCBR0 - TCBR127)	32	W	0000_0000h
400h - 7FCh	Transmit Data Burst (TDBR0 - TDBR255)	32	W	0000_0000h

66.5.5.2 Version ID (VERID)

Offset

Register	Offset
VERID	0h

Function
Indicates the version integrated for this instance on the chip and also specifies the inclusion and exclusion of several optional features.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

Field	Function
31-24 MAJOR	Major Version Number Indicates the major version number for the module specification.
23-16 MINOR	Minor Version Number Indicates the minor version number for the module specification.
15-0 FEATURE	Feature Identification Number Indicates the feature set number. 0000_0000_0000_0001b - Standard feature set 0000_0000_0000_0011b - Standard feature set with MODEM and IrDA support 0000_0000_0000_0111b - Enhanced feature set with full MODEM, IrDA, and enhanced idle detection

66.5.5.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function

Indicates the parameter configuration for this instance on the chip.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXFIFO								TXFIFO							
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

Fields

Field	Function
31-16 —	Reserved
15-8 RXFIFO	Receive FIFO Size Indicates the number of characters in the receive FIFO, which is 2^RXFIFO.
7-0 TXFIFO	Transmit FIFO Size Indicates the number of characters in the transmit FIFO, which is 2^TXFIFO.

66.5.5.4 Global (GLOBAL)

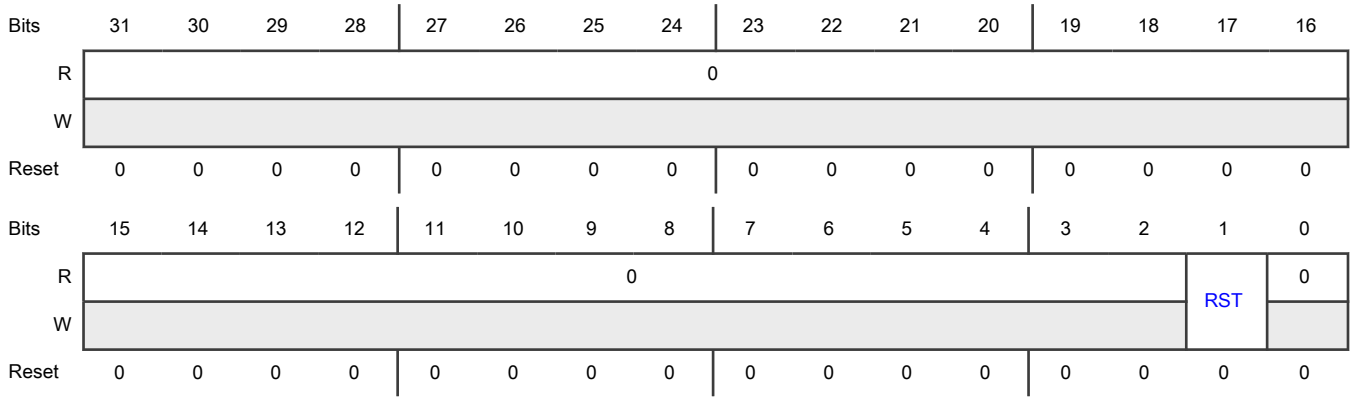
Offset

Register	Offset
GLOBAL	8h

Function

Performs global functions.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 RST	Software Reset Specifies whether the module is reset. This field resets all internal logic and registers, except Global (GLOBAL). The reset takes effect immediately and remains asserted until you negate it. There is no minimum delay required before clearing the software reset. 0b - Not reset 1b - Reset
0 —	Reserved

66.5.5.5 Pin Configuration (PINCFG)

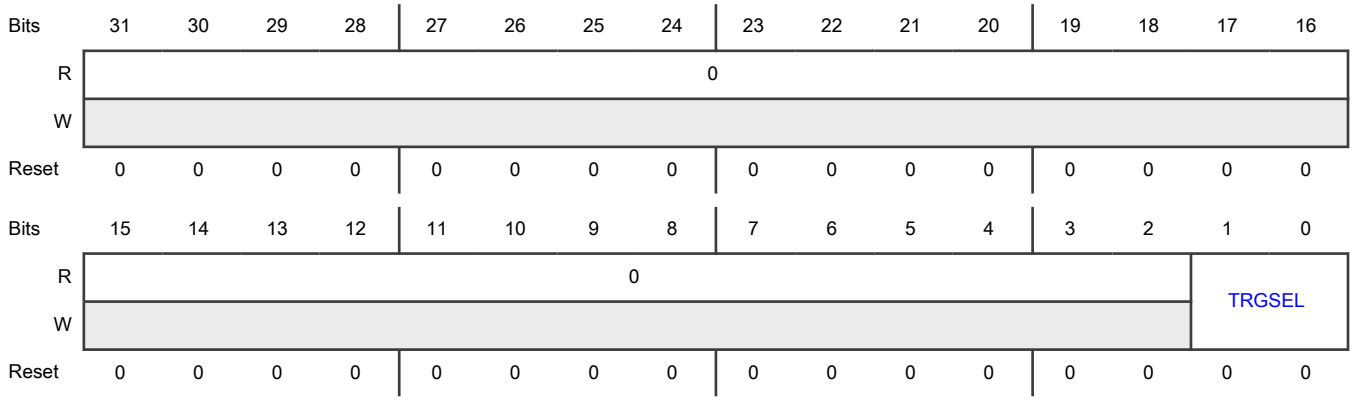
Offset

Register	Offset
PINCFG	Ch

Function

Enables the selection of input pins.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 TRGSEL	Trigger Select Configures the input trigger usage. You must change the value of this field only when both the transmitter and receiver are disabled. 00b - Input trigger disabled 01b - Input trigger used instead of the RXD pin input 10b - Input trigger used instead of the CTS_B pin input 11b - Input trigger used to modulate the TXD pin output, which (after TXINV configuration) is internally ANDed with the input trigger

66.5.5.6 Baud Rate (BAUD)

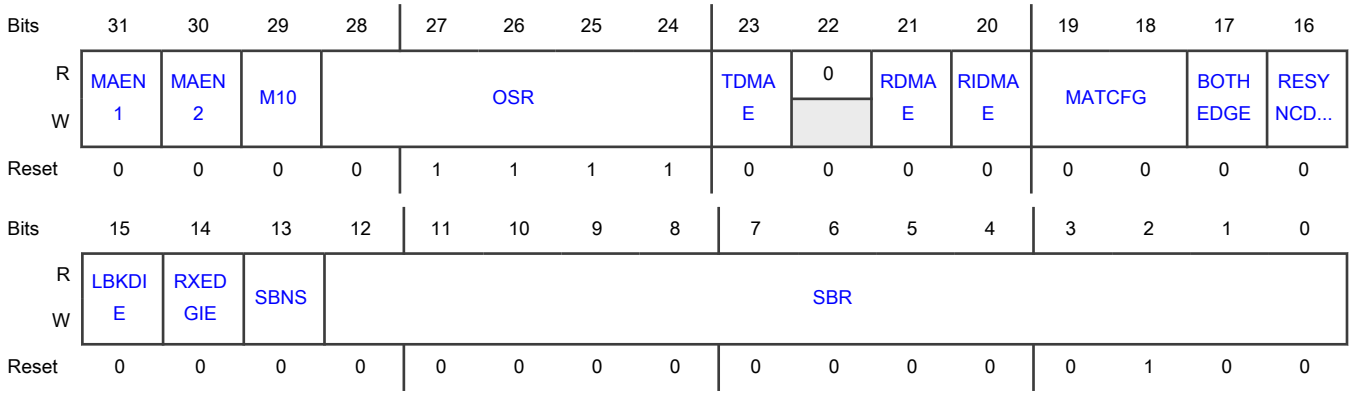
Offset

Register	Offset
BAUD	10h

Function

Configures the baud rate.

Diagram



Fields

Field	Function
31 MAEN1	Match Address Mode Enable 1 Enables automatic address matching or data matching mode for MATCH[MA1]. If this field is 0, normal operation takes place. 0b - Disable 1b - Enable
30 MAEN2	Match Address Mode Enable 2 Enables automatic address matching or data matching mode for MATCH[MA2]. If this field is 0, normal operation takes place. 0b - Disable 1b - Enable
29 M10	10-Bit Mode Select Causes the tenth bit to be a part of the serial transmission. You must change the value of this field only when both the transmitter and receiver are disabled. 0b - Receiver and transmitter use 7-bit to 9-bit data characters 1b - Receiver and transmitter use 10-bit data characters
28-24 OSR	Oversampling Ratio Configures the OSR of the receiver. You must change the value of this field only when both the transmitter and receiver are disabled. **NOTE** BAUD[OSR] results in an OSR of BAUD[OSR] + 1, for example, BAUD[OSR] = 0_0101b results in a final division by 6. 0_0000b - Results in an OSR of 16 0_0001b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0010b - Reserved 0_0011b - Results in an OSR of 4 (requires BAUD[BOTHEDGE] to be 1) 0_0100b - Results in an OSR of 5 (requires BAUD[BOTHEDGE] to be 1) 0_0101b - Results in an OSR of 6 (requires BAUD[BOTHEDGE] to be 1) 0_0110b - Results in an OSR of 7 (requires BAUD[BOTHEDGE] to be 1) 0_0111b - Results in an OSR of 8 0_1000b - Results in an OSR of 9 0_1001b - Results in an OSR of 10 0_1010b - Results in an OSR of 11 0_1011b - Results in an OSR of 12 0_1100b - Results in an OSR of 13 0_1101b - Results in an OSR of 14 0_1110b - Results in an OSR of 15 0_1111b - Results in an OSR of 16 1_0000b - Results in an OSR of 17 1_0001b - Results in an OSR of 18 1_0010b - Results in an OSR of 19 1_0011b - Results in an OSR of 20 1_0100b - Results in an OSR of 21 1_0101b - Results in an OSR of 22 1_0110b - Results in an OSR of 23 1_0111b - Results in an OSR of 24 1_1000b - Results in an OSR of 25 1_1001b - Results in an OSR of 26 1_1010b - Results in an OSR of 27 1_1011b - Results in an OSR of 28 1_1100b - Results in an OSR of 29 1_1101b - Results in an OSR of 30 1_1110b - Results in an OSR of 31 1_1111b - Results in an OSR of 32
23 TDMAE	Transmitter DMA Enable Enables STAT[TDRE] to generate a DMA request. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
22 —	Reserved
21 RDMAE	Receiver Full DMA Enable Enables STAT[RDRF] to generate a DMA request. 0b - Disable 1b - Enable
20 RIDMAE	Receiver Idle DMA Enable Enables STAT[IDLE] to generate a DMA request. If this field is 1, reading Data (DATA) when either DATA[RXEMPT] or DATA[IDLINE] is 1 generates an end-of-packet response until the completion of the DMA minor loop. During an end-of-packet response, reading Data (DATA) returns 0000_33FFh and does not pull data from the receive FIFO. STAT[IDLE] becomes 0 on completion of the minor loop, provided an end-of-packet response is generated and either the receive FIFO is empty or the receiver is active. 0b - Disable 1b - Enable
19-18 MATCFG	Match Configuration Configures the match addressing mode used. You must change the value of this field only when both the transmitter and receiver are disabled. 00b - Address match wake-up 01b - Idle match wake-up 10b - Match on and match off 11b - Enables RWU on data match and match on or off for the transmitter CTS input
17 BOTHEDGE	Both Edge Sampling Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given OSR. This field must be 1 for OSRs between x4 and x7 and is optional for higher OSRs. You must change the value of this field only when the receiver is disabled. If this field is 0, the receiver samples input data using the rising edge of the baud rate clock. If this field is 1, the receiver samples input data using the rising and falling edges of the baud rate clock. 0b - Rising edge 1b - Both rising and falling edges
16 RESYNCDIS	Resynchronization Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Disables resynchronization of the received data word when a data one followed by data zero transition is detected. You must change the value of this field only when the receiver is disabled. 0b - Enable 1b - Disable
15 LBKDIE	LIN Break Detect Interrupt Enable Enables STAT[LBKDIF] to generate hardware interrupt requests. If this field is 0, hardware interrupts from STAT[LBKDIF] (uses polling) are disabled. If this field is 1, hardware interrupts are requested when STAT[LBKDIF] is 1. 0b - Disable 1b - Enable
14 RXEDGIE	RX Input Active Edge Interrupt Enable Enables STAT[RXEDGIF] to generate interrupt requests. If this field is 0, hardware interrupts from STAT[RXEDGIF] are disabled. If this field is 1, hardware interrupts are requested when STAT[RXEDGIF] is 1. Changing the value of CTRL[LOOPS] or CTRL[RSRC] when this field (RXEDGIE) is 1 can cause STAT[RXEDGIF] to become 1. 0b - Disable 1b - Enable
13 SBNS	Stop Bit Number Select Determines whether data characters include one or two stop bits. You must change the value of this field only when both the transmitter and receiver are disabled. 0b - One stop bit 1b - Two stop bits
12-0 SBR	Baud Rate Modulo Divisor Sets the modulo divide rate for the baud rate generator. • If SBR is 0, baud rate generator is disabled. • If SBR is 1–8191, baud rate = baud clock ÷ ((OSR + 1) × SBR). You must update the 13-bit baud rate setting [SBR12:SBR0] only when both the transmitter and receiver are disabled (both CTRL[RE] and CTRL[TE] are 0).

66.5.5.7 Status (STAT)

Offset

Register	Offset
STAT	14h

Function

Provides the module status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LBKDIF	RXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W	W1C	W1C										W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MA1F	MA2F	0				TSF	MSF	0						AME	LBKFE
W	W1C	W1C														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 LBKDIF	LIN Break Detect Interrupt Flag Indicates whether a LIN break character is detected. This field becomes 1 when the LIN break detect circuitry is enabled and a LIN break character is detected. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
30 RXEDGIF	RXD Pin Active Edge Interrupt Flag Indicates whether an active edge on the receive pin has occurred.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field becomes 1 whenever the receiver is enabled and an active edge (falling if STAT[RXINV] is 0; rising if STAT[RXINV] is 1) on the RXD pin occurs. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not occurred 1b - Occurred When writing 0b - No effect 1b - Clear the flag
29 MSBF	MSB First Specifies the first bit that is transmitted after the start bit. If this field is 0, LSB (bit 0) is the first bit transmitted after the start bit (which means, the first bit received after the start bit is identified as bit 0). If this field is 1, MSB (identified as bit 9, bit 8, bit 7, or bit 6) is the first bit that is transmitted, after the start bit, depending on the settings of CTRL[M], CTRL[PE], and BAUD[M10]. Writing 1 to this field reverses the order of the bits that are transmitted and received on the wire. This field does not affect the polarity of the bits, the location of the parity bit, or the location of the start or stop bits. You must change the value of this field only when both the transmitter and receiver are disabled. 0b - LSB 1b - MSB
28 RXINV	Receive Data Inversion Specifies whether receive data is inverted. Writing 1 to this field reverses the polarity of the received data input. You must change the value of this field only when the receiver is disabled. **NOTE** Writing 1 to this field inverts the RXD input for all cases: data bits, start and stop bits, break, and idle. 0b - Inverted 1b - Not inverted
27 RWUID	Receive Wake Up Idle Detect Controls, for CTRL[RWU] on idle character detection, whether the idle character that wakes up the receiver writes 1 to STAT[IDLE]. For address match wake-up, this field controls whether STAT[IDLE] = 1 when the address does not match. You must change the value of this field only when the receiver is disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If this field is 0, during the Receive Standby state (CTRL[RWU] = 1), STAT[IDLE] does not become 1 upon detection of an idle character. During address match wake-up, STAT[IDLE] does not become 1 when an address does not match. If this field is 1, during the Receive Standby state (CTRL[RWU] = 1), STAT[IDLE] becomes 1 upon detection of an idle character. During address match wake-up, STAT[IDLE] becomes 1 when an address does not match. 0b - STAT[IDLE] does not become 1 1b - STAT[IDLE] becomes 1
26 BRK13	Break Character Generation Length Selects the longer transmitted break character length. The state of this field does not affect the detection of a framing error. You must change the value of this field only when the transmitter is disabled. You can send a break character by writing 1 to CTRL[SBK], or by writing the transmit FIFO when DATA[FRETSC] is 1 and DATA[R9T9] is 0. 0b - 9 to 13 bit times 1b - 12 to 15 bit times
25 LBKDE	LIN Break Detection Enable Enables LIN break detection. If this field is 0, LIN break detect is disabled, and only a normal break character can be detected. If this field is 1, LIN break detect is enabled and the LIN break character is detected at a length of 11 bit times (if CTRL[M] is 0), 12 bit times (if CTRL[M] is 1), or 13 bit times (if BAUD[M10] is 1). This field selects a longer break character detection length. When the field is 1, receive data is not stored in the receive FIFO. **NOTE** This field enables the LIN break detect circuit and disables writing receive data to FIFO. Therefore, it ignores all characters except a LIN break. 0b - Disable 1b - Enable
24 RAF	Receiver Active Flag Indicates whether the LPUART receiver is idle or active. This field becomes 1 when the receiver detects the beginning of a valid start bit, and the field becomes 0 automatically when the receiver detects an idle line. 0b - Idle, waiting for a start bit 1b - Receiver active (RXD pin input not idle)
23 TDRE	Transmit Data Register Empty Flag Indicates whether the transmit FIFO level is greater than, equal to, or less than the watermark.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	After the transmit FIFO is enabled, this field becomes 1 when the number of datawords in the transmit FIFO is equal to, or less than the number that WATER[TXWATER] indicates. To make the value of this field 0, write to it until the number of words in the transmit FIFO is greater than the number that WATER[TXWATER] indicates. After the transmit FIFO is disabled, this field becomes 1 to indicate that the FIFO level is less than the watermark. To make the value of this field 0 again, write to Data (DATA). This register is not affected by a character that is in the process of being transmitted; it is updated at the start of each transmitted character. 0b - Greater than watermark 1b - Equal to or less than watermark
22 TC	Transmission Complete Flag Indicates whether the transmitter is active. This field becomes 0 when a transmission is in progress or a preamble or break character is loaded; in other words, when the transmitter is active (sending data, a preamble, or a break). The field becomes 1 when the transmit buffer is empty and no data, preamble, or break character is being transmitted; in other words, when the transmission activity is complete. When this happens, the transmit data output signal becomes idle (logic 1). This field becomes 0 after you write to Data (DATA) to transmit new data, queuing a preamble by first writing 0 and then writing 1 to CTRL[TE], queuing a break character by writing 1 to CTRL[SBK]. 0b - Transmitter active 1b - Transmitter idle
21 RDRF	Receive Data Register Full Flag Indicates whether the receive FIFO level is less than, equal to, or greater than the watermark. This field becomes 1 when the number of datawords in the receive buffer is greater than the number that WATER[RXWATER] indicates and the receive FIFO is enabled. To write 0 to this field, read Data (DATA) until the number of datawords in the receive FIFO is equal to, or less than the number that WATER[RXWATER] indicates. When the receive FIFO is disabled, this field (RDRF) becomes 1 if the receive buffer (Data (DATA)) is full. To make this field 0, read Data (DATA). A character that is in the process of being received does not cause a change in this field until the entire character is received. Even if this field is 1, the character continues to be received until an overrun condition occurs after the entire character is received. 0b - Equal to or less than watermark 1b - Greater than watermark
20 IDLE	Idle Line Flag Indicates whether an idle line is detected. This field becomes 1 when the LPUART receive line becomes idle for a full character time after a period of activity. When CTRL[ILT] is 0, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bit time count towards the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. After CTRL[ILT] becomes 1, the receiver does not start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end

Table continues on the next page...

Table continued from the previous page...

Field	Function
	of the previous character do not count towards the full character time of logic high needed for the receiver to detect an idle line. For this field to become 0, write 1 to it. After the field becomes 0, you cannot write 1 to it again until after a new character is stored in the receive buffer or a LIN break character writes 1 to STAT[LBKDIF]. This field becomes 1 only once, even if the receive line remains idle for an extended period. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Idle line detected 1b - Idle line not detected When writing 0b - No effect 1b - Clear the flag
19 OR	Receiver Overrun Flag Indicates whether there is receive overrun. This field becomes 1 when you cannot prevent STAT[RDRF] from overflowing with data. The field becomes 1 immediately after the stop bit is completely received for the dataword that overflows the buffer and all the other error fields (STAT[FE], STAT[NF], and STAT[PF]) are prevented from becoming 1. The data in the shift register is lost, but the data already in the LPUART data registers is not affected. If STAT[LBKDE] is enabled and a LIN break is detected, this field becomes 1 if STAT[LBKDIF] is not 0 before the next data character is received. When this field is 1, no additional data is stored in the receive FIFO even if sufficient room exists. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No overrun 1b - Receive overrun (new LPUART data is lost) When writing 0b - No effect 1b - Clear the flag
18 NF	Noise Flag Indicates whether noise is detected in the received character of Data (DATA). The advanced sampling technique used in the receiver takes three samples in each of the received bits. If some of these samples disagree with the rest of the samples within any bit time in the frame, then noise is detected for that character. This field becomes 1 whenever the next character to be read from Data (DATA) is received with noise detected within the character.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - No noise detected 1b - Noise detected When writing 0b - No effect 1b - Clear the flag
17 FE	Framing Error Flag Indicates whether a framing error is detected. This field becomes 1 whenever the next character to be read from Data (DATA) is received with logic 0 detected where a stop bit was expected. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No framing error detected (this does not guarantee that the framing is correct) 1b - Framing error detected When writing 0b - No effect 1b - Clear the flag
16 PF	Parity Error Flag Indicates whether a parity error is detected. This field becomes 1 whenever the next character to be read from Data (DATA) is received when parity is enabled (CTRL[PE] is 1) and the parity bit in the received character does not agree with the expected parity value. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No parity error detected 1b - Parity error detected When writing 0b - No effect 1b - Clear the flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 MA1F	Match 1 Flag Indicates whether the received data is equal to MATCH[MA1]. This field becomes 1 whenever the next character to be read from Data (DATA) matches the value of MATCH[MA1]. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not equal to MA1 1b - Equal to MA1 When writing 0b - No effect 1b - Clear the flag
14 MA2F	Match 2 Flag Indicates whether the received data is equal to MATCH[MA2]. This field becomes 1 whenever the next character to be read from Data (DATA) matches the value of MATCH[MA2]. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not equal to MA2 1b - Equal to MA2 When writing 0b - No effect 1b - Clear the flag
13-10 —	Reserved
9 TSF	Timeout Status Flag Indicates whether a field in Timeout Status (TOSR) is 1 and configured to generate an interrupt. 0b - Field is 0 1b - Field is 1
8 MSF	MODEM Status Flag Indicates whether a field in MODEM Status (MSR) is 1 and configured to generate an interrupt.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Field is 0 1b - Field is 1
7-2 —	Reserved
1 AME	Address Mark Enable Configures the location of the address mark when configured for MSB first transfers. This field has no effect when configured for LSB first and you must change the value of this field only when both the transmitter and receiver are disabled. If this field is 0, address mark in character is MSB. If this field is 1, the address mark is stored in Data (DATA) at MSB (or MSB-1 when the parity bit is enabled). In other words, the address mark in character is the last bit before the stop bit (or parity bit when enabled). 0b - Disable 1b - Enable
0 LBKFE	LIN Break Flag Enable Enables the LIN break flag to assert whenever a LIN break character is detected. Unlike STAT[LBKDE], this does not impact data being stored in the receive data buffer, but does cause STAT[LBKDIF] to become 1 whenever a LIN break is detected. Because a LIN break is longer than a normal character, the LIN break triggers a write to STAT[RDRF] with the data fields as 0 and STAT[FE] as 1. The character following the LIN break has DATA[LINBRK] as 1 to indicate that the previous character was a LIN break. You must change the value of this field only when both the transmitter and receiver are disabled. If this field is 1, the LIN break character is detected at a length of 11-bit times (if CTRL[M] is 0), 12 (if CTRL[M] is 1), or 13 (if BAUD[M10] is 1). 0b - Disable 1b - Enable

66.5.5.8 Control (CTRL)

Offset

Register	Offset
CTRL	18h

Function

Controls various optional features of the LPUART system.

You must write to the fields of this register only when both the transmitter and receiver are disabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	R8T9	R9T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MA1IE	MA2IE	0	0	M7	IDLECFG			LOOPS	Reserved	RSRC	M	WAKE	ILT	PE	PT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 R8T9	Receive Bit 8 Transmit Bit 9 Contains R8 and T9 that correspond to different functions. R8 is the ninth data bit received after you configure LPUART for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading Data (DATA). T9 is the tenth data bit transmitted after you configure LPUART for 10-bit data formats. When writing 10-bit data, write T9 before writing to Data (DATA). If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, then you need not write to it each time you write to Data (DATA). **NOTE** R8 is a read-only bit and T9 is a write-only bit; the value read is different from the value written.
30 R9T8	Receive Bit 9 Transmit Bit 8 Contains R9 and T8 that correspond to different functions. R9 is the tenth data bit received after you configure LPUART for 10-bit data formats. When reading 10-bit data, read R9 before reading Data (DATA). T8 is the ninth data bit transmitted after you configure LPUART for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing to Data (DATA). If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, then you need not write to it each time you write to Data (DATA). **NOTE** R9 is a read-only field and T8 is a write-only field; the value read is different from the value written.
29 TXDIR	TXD Pin Direction in Single-Wire Mode Determines the direction of data at the TXD pin, in Single-Wire mode, when LPUART is configured for a single-wire half-duplex operation (CTRL[LOOPS] and CTRL[RSRC] are 1). When writing 0 to this field, the transmitter finishes transmitting the current character (if any) before the receiver starts receiving data from the TXD pin.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Input 1b - Output
28 TXINV	Transmit Data Inversion Specifies whether transmit data is inverted. Writing 1 to this field reverses the polarity of the transmitted data output. This action inverts the TXD output for all cases: data bits, start and stop bits, break, and idle. 0b - Not inverted 1b - Inverted
27 ORIE	Overrun Interrupt Enable Enables STAT[OR] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when OR interrupts are disabled. 0b - Disable 1b - Enable
26 NEIE	Noise Error Interrupt Enable Enables STAT[NF] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when NF interrupts are disabled. 0b - Disable 1b - Enable
25 FEIE	Framing Error Interrupt Enable Enables STAT[FE] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when FE interrupts are disabled. 0b - Disable 1b - Enable
24 PEIE	Parity Error Interrupt Enable Enables STAT[PF] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when PF interrupts are disabled. 0b - Disable 1b - Enable
23 TIE	Transmit Interrupt Enable Enables STAT[TDRE] to generate interrupt requests if STAT[TDRE] is 1. 0b - Disable 1b - Enable
22	Transmission Complete Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TCIE	Enables STAT[TC] to generate interrupt requests if STAT[TC] is 1. 0b - Disable 1b - Enable
21 RIE	Receiver Interrupt Enable Enables STAT[RDRF] to generate hardware interrupt requests if STAT[RDRF] is 1. 0b - Disable 1b - Enable
20 ILIE	Idle Line Interrupt Enable Enables hardware interrupts. This field enables STAT[IDLE] to generate interrupt requests. If this field is 0, hardware interrupts from STAT[IDLE] are disabled and polling is used, and if this field is 1, hardware interrupts are enabled when STAT[IDLE] is 1. 0b - Disable 1b - Enable
19 TE	Transmitter Enable Enables the LPUART transmitter. Using this field, you can also queue an idle preamble by first writing 0 and then writing 1 to this field. After this field becomes 0, the field reads 1 until the transmitter has completed the current character and the TXD pin is tristated. You can also queue a single idle character by writing to the transmit FIFO with DATA[FRETSC] and DATA[R9T9] = 1. 0b - Disable 1b - Enable
18 RE	Receiver Enable Enables the LPUART receiver. After you write 0 to this field, this field remains 1 until the receiver finishes receiving the current character (if any). 0b - Disable 1b - Enable
17 RWU	Receiver Wake-Up Control Specifies whether the LPUART receiver in standby is waiting for a wake-up condition. You can write 1 to this field to place the LPUART receiver in a Standby state. The field becomes 0 automatically when an RWU event occurs, that is, in case of an idle event when CTRL[WAKE] is 0 or an address match when CTRL[WAKE] is 1 and STAT[RWUID] is 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE You must write 1 to this field only when CTRL[WAKE] is 0 (wake-up on idle), if the channel is currently not idle. You can determine this by the value of STAT[RAF]. If the field is 1 to wake up an idle event and the channel is already idle, LPUART, possibly, discards the data. This is because the data must be received or a LIN break is detected after an Idle condition is detected before the IDLE flag is allowed to be reasserted. 0b - Normal receiver operation 1b - LPUART receiver in standby, waiting for a wake-up condition
16 SBK	Send Break Specifies whether queue break character(s) are to be sent. Writing 1 and then 0 to this field queues a break character in the transmit data stream. Additional break characters of 9 to 13 bits, or 12 to 15 bits if STAT[BRK13] is 1, and bit times of logic 0 are queued as long as this field is 1. Depending on the timing when this field is 1 and 0, relative to the character currently being transmitted, a second break character may be queued before you write 0 to this field. If the time taken to write 0 to this field is too long, for example, if the field does not become 0 by the end of the first break character, a second break character is sent. This is compared to queuing a break character through the transmit FIFO that guarantees only one break character is sent. You can also queue a single break character by writing to the transmit FIFO when DATA[FRETSC] is 1 and DATA[R9T9] is 0. 0b - Normal transmitter operation 1b - Queue break character(s) to be sent
15 MA1IE	Match 1 (MA1F) Interrupt Enable Enables the MA1F interrupt. 0b - Disable 1b - Enable
14 MA2IE	Match 2 (MA2F) Interrupt Enable Enables the MA2F interrupt. 0b - Disable 1b - Enable
13 —	Reserved
12 —	Reserved
11 M7	7-Bit Mode Select Specifies the data characters that the receiver and transmitter use.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	You must change the value of this field only after both the transmitter and receiver are disabled. 0b - 8-bit to 10-bit 1b - 7-bit
10-8 IDLECFG	Idle Configuration Configures the number of idle characters that must be received before you write 1 to STAT[IDLE]. 000b - 1 001b - 2 010b - 4 011b - 8 100b - 16 101b - 32 110b - 64 111b - 128
7 LOOPS	Loop Mode Select Selects Loop mode. After this field becomes 1, the RXD pin is disconnected from LPUART and the transmitter output is internally connected to the receiver input. The transmitter and receiver must be enabled to use the loop function. In Loop mode or Single-Wire mode, the transmitter outputs are internally connected to the receiver input (see CTRL[RSRC]). 0b - Normal operation: RXD and TXD use separate pins 1b - Loop mode or Single-Wire mode
6 —	Reserved
5 RSRC	Receiver Source Select Determines the source of the receiver shift register input if CTRL[LOOPS] is 1. This field has no effect unless CTRL[LOOPS] is 1. If this field is 0, internal Loopback mode is selected. LPUART does not use the RXD pin. Additionally, the CTS_B pin is not used and internally driven by the RTS_B output. If this field is 1, single-wire LPUART mode is selected where the TXD pin is connected to the transmitter output and receiver input. 0b - Internal Loopback mode 1b - Single-wire mode
4	9-Bit Or 8-Bit Mode Select Specifies the data characters that the receiver and transmitter use.

Table continues on the next page...

Table continued from the previous page...

Field	Function
M	0b - 8-bit 1b - 9-bit
3 WAKE	Receiver Wake-Up Method Select Determines which condition wakes up LPUART when CTRL[RWU] = 1 and BAUD[MATCFG] = 0 (this field must be 1 when BAUD[MATCFG] = 11): • Address mark in the bit preceding the stop bit (or bit preceding the parity bit when parity is enabled) of the received data character • An idle condition on the receive pin input signal If this field is 0, CTRL[RWU] is configured for idle line wake-up, and if this field is 1, CTRL[RWU] is configured with address mark wake-up. 0b - Idle 1b - Mark
2 ILT	Idle Line Type Select Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or the stop bit. If the count begins after the start bit, a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. **NOTE** In case you write 1 to this field, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count. 0b - After the start bit 1b - After the stop bit
1 PE	Parity Enable Enables hardware parity generation and checking. If parity is enabled, the bit immediately before the stop bit is treated as the parity bit. 0b - Disable 1b - Enable
0 PT	Parity Type Selects the type of parity, even or odd, if parity is enabled (CTRL[PE] = 1): • Odd parity means that the total number of logic 1 bits in the data character, including the parity bit, is odd. • Even parity means that the total number of 1s in the data character, including the parity bit, is even. 0b - Even parity 1b - Odd parity

66.5.5.9 Data (DATA)

Offset

Register	Offset
DATA	1Ch

Function

Supports 8-bit, 16-bit, or 32-bit writes, each type of write performing a separate function. An 8-bit write to DATA[7:0] pushes {CTRL[R8T9], CTRL[R9T8], DATA[7:0]} the transmit FIFO with TSC clear. A 16-bit or 32-bit write pushes the data written into the FIFO and does not update the value of CTRL[R8T9] or CTRL[R9T8].

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status fields.

NOTE

Reads return the contents of the read-only receive FIFO and writes go to the write-only transmit FIFO, making this register work as a set of two separate registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NOISY	PARITYE	FRETSC	RXEMPT	IDLINE	LINBRK	R9T9	R8T8	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15 NOISY	Noisy Data Received Indicates whether the current received dataword contained in DATA[R9:R0] is received with noise. 0b - Received without noise 1b - Received with noise
14 PARITYE	Parity Error

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates whether the current received dataword contained in DATA[R9:R0] is received with a parity error. 0b - Received without a parity error 1b - Received with a parity error
13 FRETSC	Frame Error Transmit Special Character Specifies the way the dataword is received. For reads, this field indicates that the current received dataword contained in DATA[R9:R0] is received with a frame error. For writes, the field indicates that a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 indicates a break character when it is 0 and indicates an idle character when it is 1. The contents of DATA[T8:T0] must be 0. 0b - Received without a frame error on reads or transmits a normal character on writes 1b - Received with a frame error on reads or transmits an idle or break character on writes
12 RXEMPT	Receive Buffer Empty Indicates whether the receive buffer contains valid data. This field becomes 1 when there is no data in the receive buffer. The field does not consider data in the receive shift register. 0b - Valid data 1b - Invalid data and empty
11 IDLINE	Idle Line Indicates whether the receiver line was idle before receiving the character in DATA[9:0]. It can be read as “1” on the first character when the receiver is first enabled. The difference between this field and STAT[IDLE] is that, STAT[IDLE] flag does not set on an idle line after the receiver is first enabled, it needs to receive a character before it can become set, whereas this field does not have this limitation and can be set on the first character received if an idle line is detected beforehand. 0b - Not idle 1b - Idle
10 LINBRK	LIN Break Indicates whether the receiver line detected a LIN break before receiving the character in DATA[9:0]. This field requires the value of STAT[LBKDIF] to be 1. If this field is 0, the LIN break detect circuitry is disabled. 0b - Not detected 1b - Detected
9 R9T9	Read receive FIFO bit 9 or write transmit FIFO bit 9
8	Read receive FIFO bit 8 or write transmit FIFO bit 8

Table continues on the next page...

Table continued from the previous page...

Field	Function
R8T8	
7 R7T7	Read receive FIFO bit 7 or write transmit FIFO bit 7
6 R6T6	Read receive FIFO bit 6 or write transmit FIFO bit 6
5 R5T5	Read receive FIFO bit 5 or write transmit FIFO bit 5
4 R4T4	Read receive FIFO bit 4 or write transmit FIFO bit 4
3 R3T3	Read receive FIFO bit 3 or write transmit FIFO bit 3
2 R2T2	Read receive FIFO bit 2 or write transmit FIFO bit 2
1 R1T1	Read receive FIFO bit 1 or write transmit FIFO bit 1
0 R0T0	Read receive FIFO bit 0 or write transmit FIFO bit 0

66.5.5.10 Match Address (MATCH)

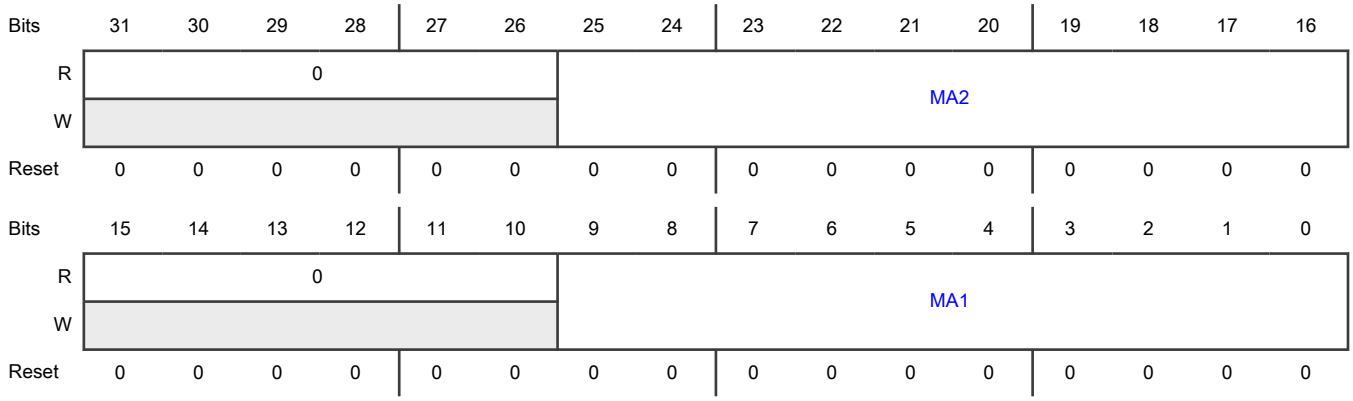
Offset

Register	Offset
MATCH	20h

Function

Provides addresses for address matching during the receiver operation.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 MA2	Match Address 2 Is compared to input data addresses when the most significant bit is 1 and the associated Baud Rate (BAUD) field is 1. If a match occurs, the data that follows is transferred to Data (DATA). If a match fails, the data that follows is discarded. You must write to MATCH[MA1] and MATCH[MA2] only when the associated Baud Rate (BAUD) field is 0.
15-10 —	Reserved
9-0 MA1	Match Address 1 Is compared to input data addresses when the most significant bit is 1 and the associated Baud Rate (BAUD) field is 1. If a match occurs, the data that follows is transferred to Data (DATA). If a match fails, the data that follows is discarded. You must write to MATCH[MA1] and MATCH[MA2] fields only when the associated field in Baud Rate (BAUD) is 0.

66.5.5.11 MODEM IrDA (MODIR)

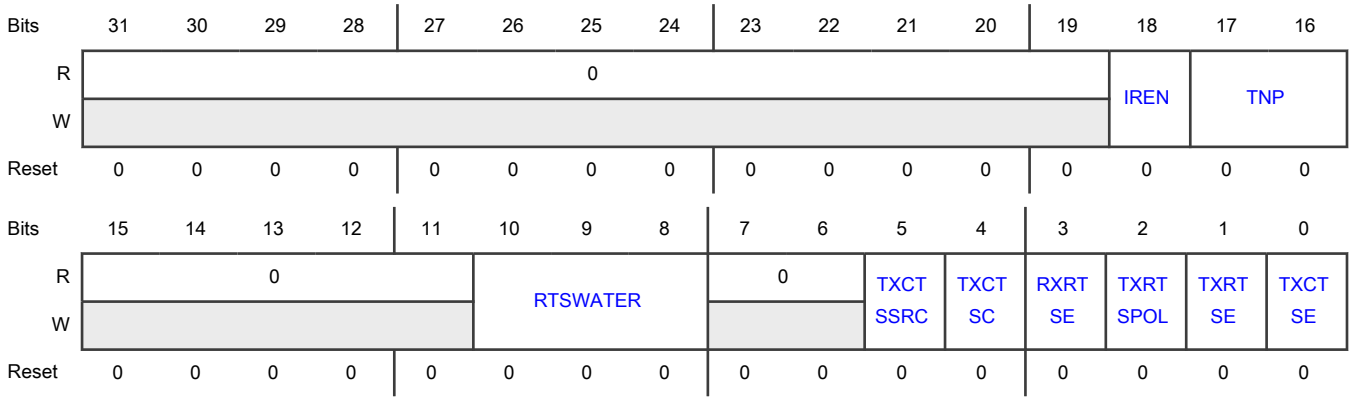
Offset

Register	Offset
MODIR	24h

Function

Controls options for setting the MODEM configuration.

Diagram



Fields

Field	Function
31-19 —	Reserved
18 IREN	IR Enable Enables IR modulation and demodulation. You must change the value of this field only when both the transmitter and receiver are disabled. 0b - Disable 1b - Enable
17-16 TNP	Transmitter Narrow Pulse Specifies whether LPUART transmits a 1 ÷ OSR, 2 ÷ OSR, 3 ÷ OSR, or 4 ÷ OSR narrow pulse when the IR pulse is enabled. You must change the value of this field only when both the transmitter and receiver are disabled. The IR pulse width must be configured to less than half of the OSR. Common pulse widths are 3 ÷ 16, 1 ÷ 16, 1 ÷ 32, or 1 ÷ 4 of the bit length. You can configure these by selecting the appropriate OSR and pulse width. 00b - 1 ÷ OSR 01b - 2 ÷ OSR 10b - 3 ÷ OSR 11b - 4 ÷ OSR
15-11 —	Reserved
10-8 RTSWATER	Receive RTS Configuration Configures the assertion and negation of the receiver's RTS_B output. The receiver's RTS_B output negates when the number of empty words in the receive FIFO is greater or equal to the value of this field. If this field is 0, the RTS_B pin negates when the receive FIFO is full. For the

Table continues on the next page...

Table continued from the previous page...

Field	Function
	purpose of receive RTS_B generation, the number of words in the receive FIFO updates when a start bit is detected. This supports additional latency between RTS_B negation and the external transmitter ceasing transmission. If both receive RTS_B and address or data matching is enabled, RTS_B could assert at the end of a character if there exists no match. You must change the value of this field only when the receiver is disabled.
7-6 —	Reserved
5 TXCTSSRC	Transmit CTS Source Configures the source of the CTS input. 0b - The CTS_B pin 1b - An internal connection to the receiver address match result
4 TXCTSC	Transmit CTS Configuration Configures whether the CTS state or input is checked or sampled at the start of each character or only when the transmitter is idle. 0b - Sampled at the start of each character 1b - Sampled when the transmitter is idle
3 RXRTSE	Receiver RTS Enable Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. You must change the value of this field only when the receiver is disabled. If this field is 0, the receiver has no effect on RTS. If this field is 1, RTS is deasserted if STAT[RDRF] is 1 or a start bit is detected that causes STAT[RDRF] to become 1. RTS is asserted if STAT[RDRF] is 0 and has not detected a start bit that causes STAT[RDRF] to become 1. **NOTE** Do not write 1 to both MODIR[RXRTSE] and MODIR[TXRTSE]. 0b - Disable 1b - Enable
2 TXRTSPOL	Transmitter RTS Polarity Controls the polarity of the transmitter RTS. This field does not affect the polarity of the receiver RTS that remains negated in the active-low state unless MODIR[TXRTSE] is 1. You must change the value of this field only when the transmitter is disabled. 0b - Active low 1b - Active high
1	Transmitter RTS Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TXRTSE	Controls the operation of RTS before and after a transmission. You must change the value of this field only when the transmitter is disabled. If this field is 0, the transmitter has no effect on RTS, and if this field is 1, a character is placed into an empty transmit shift register. RTS asserts 1-bit time before the start bit is transmitted and deasserts 1-bit time after all characters in the transmitter FIFO and shift register are completely sent, including the last stop bit. 0b - Disable 1b - Enable
0 TXCTSE	Transmitter CTS Enable Enables the operation of the transmitter. You can write 1 to this field irrespective of the states of MODIR[TXRTSE] and MODIR[RXRTSE]. If this field is 1, the transmitter checks the state of the CTS signal each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the TXD signal remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS, when a character is being sent, do not affect its transmission. 0b - Disable 1b - Enable

66.5.5.12 FIFO (FIFO)

Offset

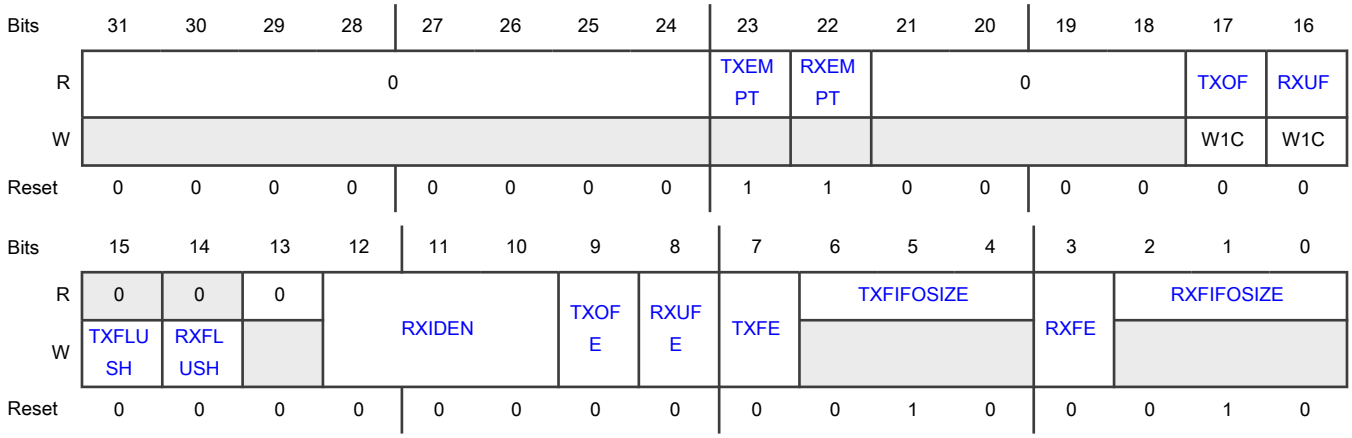
Register	Offset
FIFO	28h

Function

Provides you the ability to turn on and turn off the FIFO functionality.

This register also provides you the size of the FIFO that has been implemented. You can read this register at any time and must write to it only when CTRL[RE] and CTRL[TE] are 0 and the FIFO is empty.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 TXEMPT	Transmit FIFO Or Buffer Empty Indicates whether the transmit buffer is empty. This field becomes 1 when there is no data in the transmit FIFO or buffer. The field does not consider data in the transmit shift register. 0b - Not empty 1b - Empty
22 RXEMPT	Receive FIFO Or Buffer Empty Indicates whether the receive buffer is empty. This field becomes 1 when there is no data in the receive FIFO or buffer. The field does not consider data in the receive shift register. 0b - Not empty 1b - Empty
21-18 —	Reserved
17 TXOF	Transmitter FIFO Overflow Flag Indicates whether more data has been written to the transmit FIFO than it can hold. If this field is 0, no transmit FIFO overflow has occurred since the last time the field was cleared, and if this field is 1, at least one transmit FIFO overflow has occurred since the last time the field was cleared. This field becomes 1 regardless of the value of FIFO[TXOFE]. However, an interrupt is issued to the host only if FIFO[TXOFE] is 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - No overflow 1b - Overflow When writing 0b - No effect 1b - Clear the flag
16 RXUF	Receiver FIFO Underflow Flag Indicates whether more data has been read from the receive FIFO than was present. If this field is 0, no receive FIFO underflow has occurred since the last time the field was cleared, and if this field is 1, at least one receive FIFO underflow has occurred since the last time the field was cleared. This field becomes 1 regardless of the value of FIFO[RXUFE]. However, an interrupt is issued to the host only if FIFO[RXUFE] is 1. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No underflow 1b - Underflow When writing 0b - No effect 1b - Clear the flag
15 TXFLUSH	Transmit FIFO Flush Causes all data that is stored in the transmit FIFO to be flushed. If you write 0 to this field, no flush operation occurs, and if you write 1 to this field, all data in the transmit FIFO or buffer clears out. This does not affect data in the transmit shift register. 0b - No effect 1b - All data flushed out
14 RXFLUSH	Receive FIFO Flush Causes all data that is stored in the receive FIFO to be flushed. If you write 0 to this field, no flush operation occurs, and if you write 1 to this field, all data in the receive FIFO or buffer clears out.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This does not affect data in the receive shift register. 0b - No effect 1b - All data flushed out
13 —	Reserved
12-10 RXIDEN	Receiver Idle Empty Enable Enables STAT[RDRF] to become 1 when the receiver is idle for a number of idle characters and the FIFO is not empty. This feature is not supported when the receiver extended idle time is enabled. 000b - Disable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle 001b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for one character 010b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for two characters 011b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for four characters 100b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for eight characters 101b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for 16 characters 110b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for 32 characters 111b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for 64 characters
9 TXOFE	Transmit FIFO Overflow Interrupt Enable Enables FIFO[TXOF] to generate an interrupt to the host. 0b - Disable 1b - Enable
8 RXUFE	Receive FIFO Underflow Interrupt Enable Enables FIFO[RXUF] to generate an interrupt to the host. 0b - Disable 1b - Enable
7 TXFE	Transmit FIFO Enable Enables the transmit FIFO. If this field is 0, the transmit buffer operates as a FIFO of depth equal to 1 dataword, regardless of the value in FIFO[TXFIFOSIZE]. Both CTRL[TE] and CTRL[RE] must be 0 before you change the value of this field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If this field is 1, the built-in FIFO structure for the transmit buffer is enabled. FIFO[TXFIFOSIZE] indicates the size of the FIFO structure. 0b - Disable 1b - Enable
6-4 TXFIFOSIZE	Transmit FIFO Buffer Depth Indicates the maximum number of transmit datawords (transmit FIFO buffer depth) that can be stored in the transmit buffer. 000b - 1 001b - 4 010b - 8 011b - 16 100b - 32 101b - 64 110b - 128 111b - 256
3 RXFE	Receive FIFO Enable Enables the receive FIFO. If this field is 0, the receive buffer operates as a FIFO of depth equal to 1 dataword, regardless of the value in FIFO[RXFIFOSIZE]. Both CTRL[RE] and CTRL[TE] must be 0 before you change the value of this field. If this field is 1, the built-in FIFO structure for the receive buffer is enabled. FIFO[RXFIFOSIZE] indicates the size of the FIFO structure. 0b - Disable 1b - Enable
2-0 RXFIFOSIZE	Receive FIFO Buffer Depth Indicates the maximum number of receive datawords (receive FIFO buffer depth) that can be stored in the receive buffer before an overrun occurs. 000b - 1 001b - 4 010b - 8 011b - 16 100b - 32 101b - 64 110b - 128 111b - 256

66.5.5.13 Watermark (WATER)

Offset

Register	Offset
WATER	2Ch

Function

Provides the ability to set a programmable threshold for notification, or sets the programmable thresholds to indicate that transmit data can be written or receive data can be read.

You may read this register at any time but must write to it only when CTRL[TE] is 0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				RXCOUNT				0				RXWATER			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				TXCOUNT				0				TXWATER			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 RXCOUNT	Receive Counter Indicates the number of datawords in the receive FIFO or buffer. If a dataword is being received in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate the room left in the receive FIFO or buffer.
23-19 —	Reserved
18-16 RXWATER	Receive Watermark Generates an interrupt or a DMA request if the number of datawords in the receive FIFO or buffer is greater than the value of this field. For proper operation, the value of this field must be less than the size of the receive FIFO or buffer, as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE].

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-12 —	Reserved
11-8 TXCOUNT	Transmit Counter Indicates the number of datawords in the transmit FIFO or buffer. If a dataword is being transmitted to the transmit shift register, it is not included in the count. This value may be used in conjunction with the value of FIFO[TXFIFOSIZE] to calculate the room left in the transmit FIFO or buffer.
7-3 —	Reserved
2-0 TXWATER	Transmit Watermark Generates an interrupt or a DMA request when the number of datawords in the transmit FIFO or buffer is equal to or less than the value of this field. For proper operation, the value of this field must be less than the size of the transmit buffer or FIFO, as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE].

66.5.5.14 Data Read-Only (DATARO)

Offset

Register	Offset
DATARO	30h

Function
Indicates the first entry in the receive FIFO, but does not pull data from the FIFO.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA															
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-0 DATA	Receive Data Indicates the first entry from the receive FIFO. This register has the same functionality as that of Data (DATA).

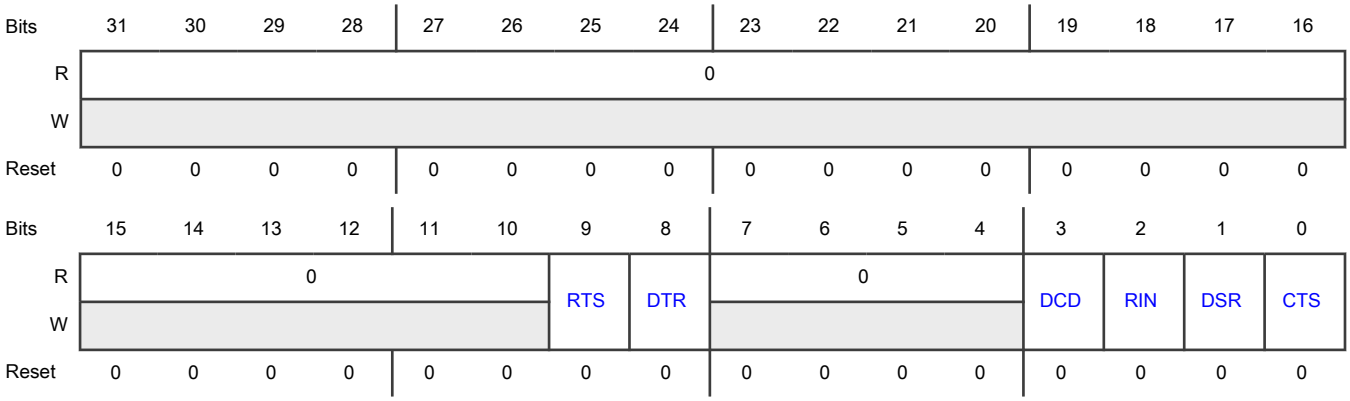
66.5.5.15 MODEM Control (MCR)

Offset

Register	Offset
MCR	40h

Function
Controls the operation of the MODEM pins.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 RTS	Request To Send Configures the default state of the RTS_B pin when the function is disabled. 0b - Logic one

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Logic zero
8 DTR	Data Terminal Ready Configures the default state of the DTR_B pin. 0b - Logic one 1b - Logic zero
7-4 —	Reserved
3 DCD	Data Carrier Detect Configures the interrupt, DCD_B, for change of state of the DCD_B pin. 0b - Disable interrupt 1b - Enable interrupt
2 RIN	Ring Indicator Configures the interrupt, RIN_B, for change of state on the RIN_B pin. 0b - Disable interrupt 1b - Enable interrupt
1 DSR	Data Set Ready Configures the interrupt, DSR_B, for change of state on the DSR_B pin. 0b - Disable interrupt 1b - Enable interrupt
0 CTS	Clear To Send Configures the interrupt, CTS_B, for change of state on the CTS_B pin. 0b - Disable interrupt 1b - Enable interrupt

66.5.5.16 MODEM Status (MSR)

Offset

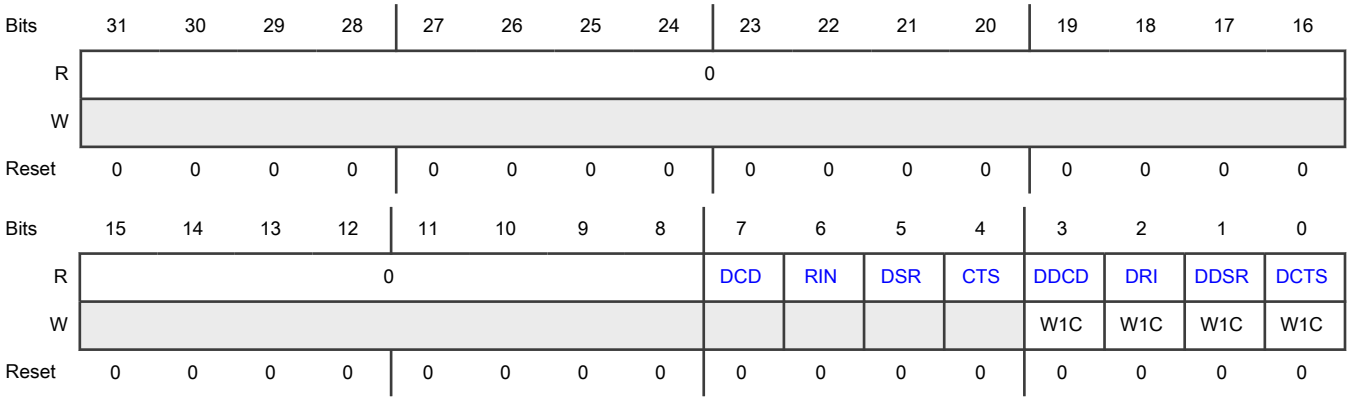
Register	Offset
MSR	44h

Function
Indicates the status of the MODEM pins.

NOTE

You must appropriately configure the PAD connecting to DCD_B, RIN_B, DSR_B, and CTS_B inputs to get appropriate reset values for the MSR register fields.

Diagram



Fields

Field	Function
31-8 —	Reserved
7 DCD	Data Carrier Detect Indicates the state of the DCD_B pin. 0b - Logic one 1b - Logic zero
6 RIN	Ring Indicator Indicates the state of the RIN_B pin. 0b - Logic one 1b - Logic zero
5 DSR	Data Set Ready Indicates the state of the DSR_B pin. 0b - Logic one 1b - Logic zero
4 CTS	Clear To Send Indicates the state of the CTS_B pin. 0b - Logic one 1b - Logic zero

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 DDCD	Delta Data Carrier Detect Indicates whether the DCD_B pin changed state since the last time this field was 0. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not change state 1b - Changed state When writing 0b - No effect 1b - Clear the flag
2 DRI	Delta Ring Indicator Indicates whether the RIN_B pin changed state since the last time this field was 0. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not change state 1b - Changed state When writing 0b - No effect 1b - Clear the flag
1 DDSR	Delta Data Set Ready Indicates whether the DSR_B pin changed state since the last time this field was 0. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not change state 1b - Changed state When writing 0b - No effect 1b - Clear the flag
0 DCTS	Delta Clear To Send Indicates whether the CTS_B pin changed state since the last time this field was 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - Did not change state 1b - Changed state When writing 0b - No effect 1b - Clear the flag

66.5.5.17 Receiver Extended Idle (REIR)

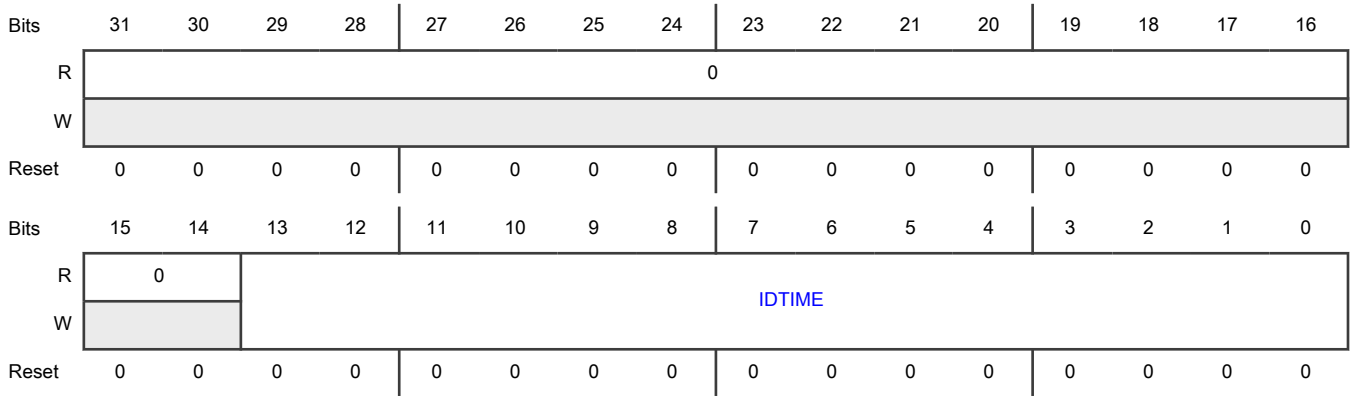
Offset

Register	Offset
REIR	48h

Function

Configures the receiver extended idle functionality. You must not change this value when the receiver is enabled.

Diagram



Fields

Field	Function
31-14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-0 IDTIME	Idle Time Configures the idle length in number of bits (baud rate) since the end of the last stop bit. This affects the behavior of the idle wake-up, STAT[IDLE], DATA[IDLINE], and STAT[RAF]. The minimum supported extended idle time is equal to one idle character. The extended idle feature is disabled when this field is 0.

66.5.5.18 Transmitter Extended Idle (TEIR)

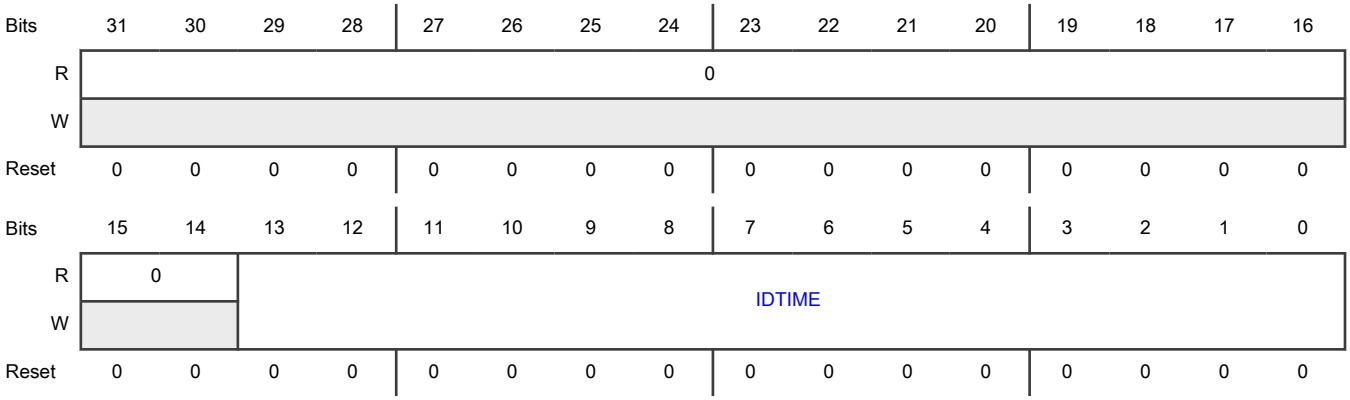
Offset

Register	Offset
TEIR	4Ch

Function

Configures the transmitter extended idle functionality. You must not change this value when the transmitter is enabled.

Diagram



Fields

Field	Function
31-14 —	Reserved
13-0 IDTIME	Idle Time Configures the transmitter idle time in number of bits (baud rate) whenever an idle character is queued through the transmit FIFO. An idle character is not automatically queued whenever the transmitter is enabled. The minimum supported extended idle time equals one idle character. The extended idle feature is disabled when this field is 0.

66.5.5.19 Half Duplex Control (HDCR)

Offset

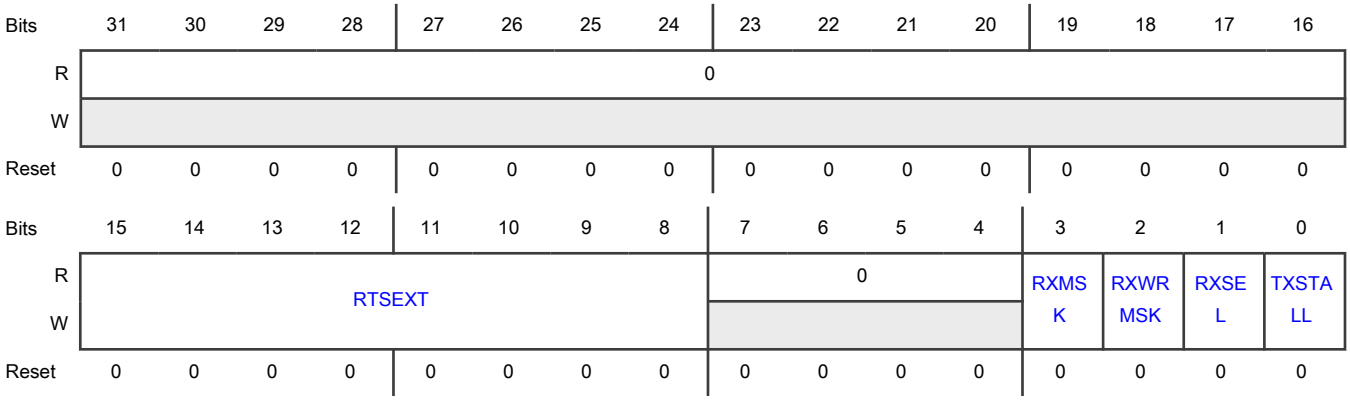
Register	Offset
HDCR	50h

Function

Provides control for half-duplex-related operations.

You can use this register instead of CTRL[LOOPS], CTRL[RSRC], and CTRL[TXDIR] functions, although you can use CTRL[LOOPS] to loop-back the transmitter outputs to the receiver.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 RTSEXT	RTS Extended Specifies RTS extension. The transmit RTS_B remains asserted for (RTSEXT + 1) bit times after the end of the last stop bit. This applies even when the transmitter's RTS_B output is disabled and is only used internally to mask the receiver.
7-4 —	Reserved
3 RXMSK	Receive Mask Specifies whether the transmitter RTS_B masks the receive data pin. When enabled, the transmitter RTS_B masks the receive data pin. 0b - Do not mask

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Mask
2 RXWRMSK	Receive FIFO Write Mask Specifies whether the transmitter RTS_B masks writes to the receive FIFO. When enabled, the transmitter RTS_B masks receive FIFO writes, but the idle flag is not affected. 0b - Do not mask 1b - Mask
1 RXSEL	Receive Select Specifies the receive data pin. When enabled, the receive data is sourced from the TXD pin. 0b - RXD 1b - TXD
0 TXSTALL	Transmit Stall Specifies whether the transmitter becomes busy when the receiver is active. When enabled, the transmitter does not become busy or asserts transmitter RTS_B when the receiver is active (STAT[RAF] is 1). 0b - No effect 1b - Does not become busy

66.5.5.20 Timeout Control (TOCR)

Offset

Register	Offset
TOCR	58h

Function

Configures the behavior of the timeout logic. Timeouts 0 and 1 are used to monitor the receiver and timeouts 2 and 3 are used to monitor the transmitter.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				TOIE				0				TOEN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11-8 TOIE	Timeout Interrupt Enable Enables the corresponding timeout flag to generate an interrupt.
7-4 —	Reserved
3-0 TOEN	Timeout Enable Enables the corresponding timeout counter.

66.5.5.21 Timeout Status (TOSR)

Offset

Register	Offset
TOSR	5Ch

Function

Indicates the status of the timeout logic. Timeouts 0 and 1 are used to monitor the receiver and timeouts 2 and 3 are used to monitor the transmitter.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				TOF				0				TOZ			
W					W1C											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Fields

Field	Function
31-12 —	Reserved
11-8 TOF	Timeout Flag Indicates whether the corresponding timeout occurred. The timeout counter is disabled when this field is 1. **NOTE** This field behaves differently for register reads and writes. When reading 0000b - Not occurred 0001b - Occurred When writing 0000b - No effect 0001b - Clear the flag
7-4 —	Reserved
3-0 TOZ	Timeout Zero Indicates whether the corresponding timeout counter equals 0.

66.5.5.22 Timeout N (TIMEOUT0 - TIMEOUT3)

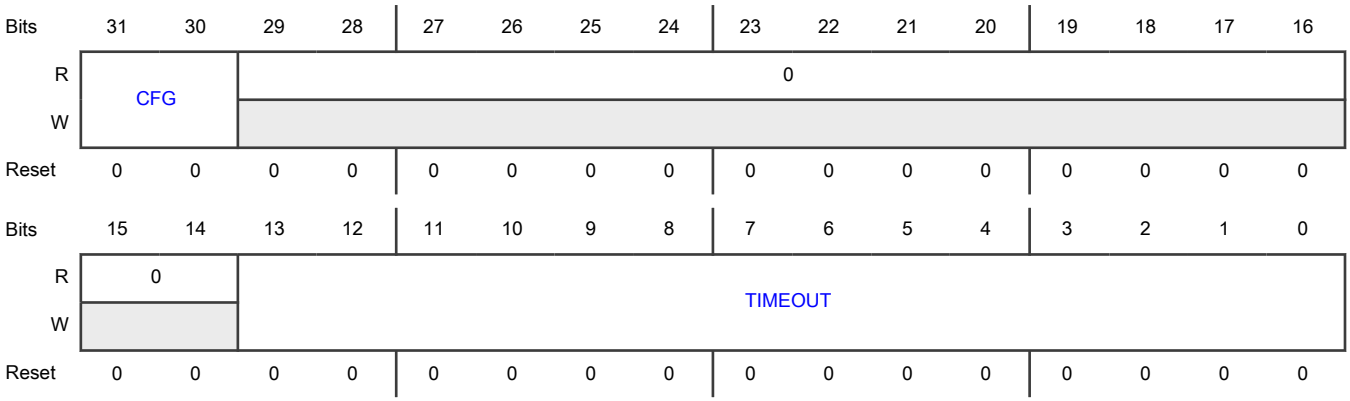
Offset

Register	Offset
TIMEOUT0	60h
TIMEOUT1	64h
TIMEOUT2	68h
TIMEOUT3	6Ch

Function

Configures the corresponding timeout counter and status field. Timeouts 0 and 1 are used to monitor the receiver and timeouts 2 and 3 are used to monitor the transmitter. You must write to this register only when the corresponding timeout is disabled or when the value of the corresponding timeout field is 1.

Diagram



Fields

Field	Function
31-30 CFG	Idle Configuration Configures the behavior of TIMEOUT[TIMEOUT]. 00b - Becomes 1 after timeout characters are received 01b - Becomes 1 when idle for timeout bit clocks 10b - Becomes 1 when idle for timeout bit clocks following the next character 11b - Becomes 1 when idle for at least timeout bit clocks, but a new character is detected before the extended idle timeout is reached
29-14 —	Reserved
13-0 TIMEOUT	Timeout Value Configures the timeout value.

66.5.5.23 Transmit Command Burst (TCBR0 - TCBR127)

Offset

For a = 0 to 127:

Register	Offset
TCBRa	200h + (a × 4h)

Function

Acts as an alias of Data (DATA), designed to support incrementing burst transfers to the transmit FIFO by a DMA controller, using aligned 8-bit, 16-bit, or 32-bit writes. The size of this register is 512 bytes:

- An aligned 32-bit write in this region pushes one entry into the transmit FIFO.
- An aligned 16-bit write in this region to TCBR*x*[15:0] pushes one entry into the transmit FIFO.
- An 8-bit write in this region to TCBR*x*[7:0] updates DATA[7:0], but does not push the data into the transmit FIFO.
- An 8-bit write in this region to TCBR*x*[15:8] pushes the data written to DATA[15:8] plus the previously written DATA[7:0] into the transmit FIFO.
- An 8-bit or 16-bit write in this region to TXBR*x*[31:16] is ignored.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-0 DATA	Data Enables writing data to Data (DATA).

66.5.5.24 Transmit Data Burst (TDBR0 - TDBR255)

Offset

For a = 0 to 255:

Register	Offset
TDBRa	400h + (a × 4h)

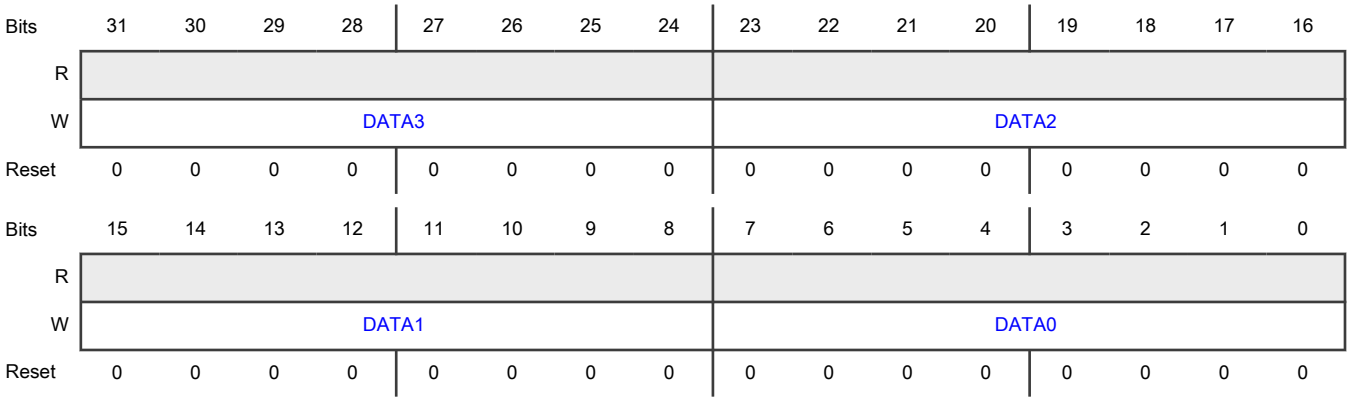
Function

Acts as an alias of Data (DATA), designed to support incrementing burst transfers to the transmit FIFO by a DMA controller using 8-bit, 16-bit, or 32-bit writes. The size of this register is 1024 bytes:

- An aligned 32-bit write in this region pushes four DATA byte entries into the transmit FIFO (DATA0 byte first). The register access is extended by three wait states.
- An aligned 16-bit write in this region to either half of a 32-bit word pushes two DATA byte entries into the transmit FIFO (DATA0 or DATA2 first). The register access is extended by one wait state.
- An 8-bit write in this region pushes one DATA byte entry into the transmit FIFO.

Byte writes to Data (DATA) use the contents of CTRL[R9T8] for the ninth data bit, CTRL[R8T9] for the tenth data bit, and zero extend DATA[FRETSC].

Diagram



Fields

Field	Function
31-24 DATA3	Data3 Enables writing of data to Data (DATA).
23-16 DATA2	Data2 Enables writing of data to Data (DATA).
15-8 DATA1	Data1 Enables writing of data to Data (DATA).

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 DATA0	Data0 Enables writing of data to Data (DATA).

Chapter 67
Synchronous Audio Interface (SAI)

67.1 Chip-specific SAI information

Table 580. Reference links to related information

Topic	Related module	Reference
Full description	SAI	SAI
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

67.1.1 Module instances

This device has two instances of the SAI module, SAI0 and SAI1.

67.1.2 SAI0/1 TCR2, RCR2, and MCR configuration

Table 581. SAI0/1 configuration

Instance	Register bit	Configuration
SAI0	TCR2[MSEL], RCR2[MSEL]	00 - SAI0 Bus Clock 01 - SAI0 MCLK output 10 - SAI1 MCLK output 11 - Reserved
SAI1	TCR2[MSEL], RCR2[MSEL]	00 - SAI1 Bus Clock 01 - SAI1 MCLK output 10 - SAI0 MCLK output 11 - Reserved
SAI0	MCR[MSEL]	SAI MCR register generates SAI MCLK output. 00 - SYSCON SAI0CLKDIV output 01 - Reserved 10 - SAI1 MCLK output 11 - Reserved
SAI1	MCR[MSEL]	SAI MCR register generates SAI MCLK output. 00 - SYSCON SAI1CLKDIV output 01 - Reserved 10 - SAI0 MCLK output 11 - Reserved

SAI1 bit clock, frame sync, and transmitter/receiver enable are synchronous with SAI0 in multiple SAI synchronous mode.

67.2 Overview

SAI provides an interface that supports full-duplex serial digital audio interfaces with frame synchronization formats such as I^{2}S, AC97, TDM, and Codec/DSP interfaces.

67.2.1 Block diagram

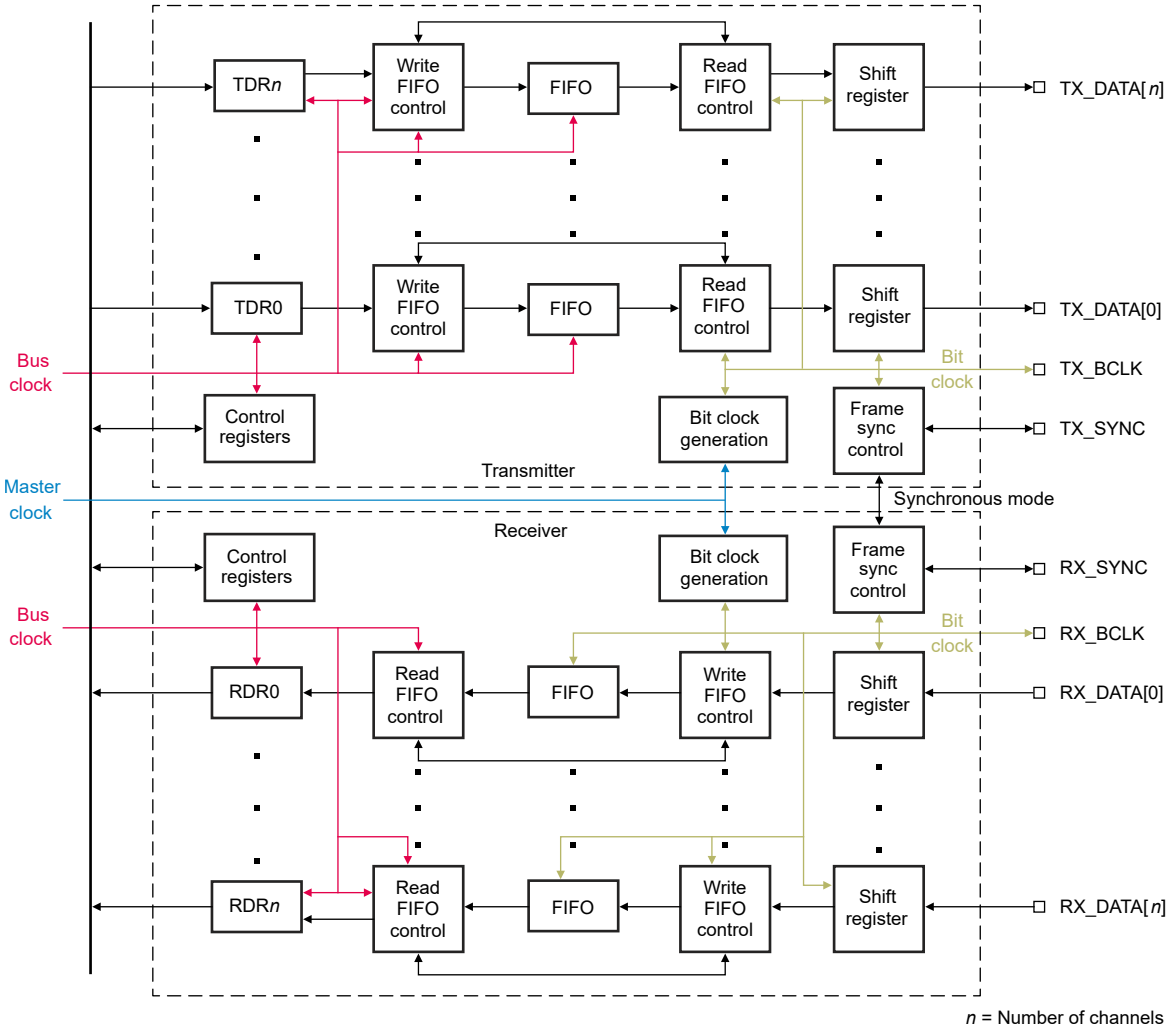


Figure 390. Block diagram

67.2.2 Features

Features can vary among chips and among SAI modules on the same chip. See the chip-specific SAI information at the beginning of this chapter.

- Transmitter with independent bit clock and frame synchronization supporting 2 data lines
- Receiver with independent bit clock and frame synchronization supporting 2 data lines
- Maximum frame size of 32 words per data line
- Word length of 8 to 32 bits

- Word length configured separately for the first word and remaining words in a frame
- Asynchronous 8 × 32-bit FIFO for each transmit and receive data line supports:
  - Graceful restart after FIFO error.
  - Automatic restart after FIFO error without software intervention.
  - 8-bit and 16-bit data packing into each 32-bit FIFO word.
  - Multiple-data-line FIFOs combining into single-data-line FIFO.

67.3 Functional description

This section provides a complete functional description of SAI.

67.3.1 Modes of operation

- Run mode
- Stop mode
- Debug mode

67.3.1.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

67.3.1.2 Stop mode

There are two situations in which you can configure the SAI transmitter or receiver to continue operating in Stop mode:

- The SAI transmitter or receiver uses an externally generated bit clock.
- The SAI transmitter or receiver uses an audio master clock that operates when the chip is in Stop mode.

Write 1 to TCSR[STOPE] to configure the transmitter to run in Stop mode, and write 1 to RCSR[STOPE] to configure the receiver to run in Stop mode.

The SAI transmitter and receiver can generate an asynchronous interrupt to wake the CPU from Stop mode.

When the chip enters Stop mode, if TCSR[STOPE] = 0, the transmitter is disabled after completing the current transmit frame. Similarly, when the chip enters Stop mode, if RCSR[STOPE] = 0, the receiver is disabled after completing the current receive frame. Entry into Stop mode is blocked (not acknowledged) when waiting for the transmitter and receiver to be disabled at the end of the current frame.

67.3.1.3 Debug mode

You can configure either or both the SAI transmitter and receiver to continue operating in Debug mode:

- Write 1 to TCSR[DBGE] to configure the transmitter to run in Debug mode.
- Write 1 to RCSR[DBGE] to configure the receiver to run in Debug mode.

When TCSR[DBGE] and RCSR[DBGE] are 0 and the chip enters Debug mode, SAI is disabled after completing the current transmit or receive frame. Debug mode does not affect the transmitter and receiver bit clocks.

67.3.2 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other. You can configure the SAI transmitter and receiver to operate with a synchronous bit clock and frame sync (see TCR2[SYNC] and RCR2[SYNC]).

If both the transmitter and receiver use the transmitter bit clock and frame sync:

- Configure the transmitter for asynchronous operation and the receiver for synchronous operation.
- Enable the receiver in Synchronous mode only after configuring both the transmitter and receiver.

- Enable the transmitter last and disable the transmitter first.

If both the transmitter and receiver use the receiver bit clock and frame sync:

- Configure the receiver for asynchronous operation and the transmitter for synchronous operation.
- Enable the transmitter in Synchronous mode only after configuring both the receiver and transmitter.
- Enable the receiver last and disable the receiver first.

When operating in Synchronous mode, the transmitter and receiver share only the bit clock, frame sync, and transmitter and receiver enable. Otherwise, the transmitter and receiver operate independently, although the configuration register settings must be consistent across both the transmitter and receiver.

67.3.2.1 Multiple SAI Synchronous mode

Some chips support synchronous operation between multiple SAI modules. This mode requires you to configure the source of the bit clock and frame sync for asynchronous operation, and to configure the remaining users of the bit clock and frame sync for synchronous operation.

Synchronous operation between multiple SAI transmitters or receivers also requires you to enable the source of the bit clock and frame sync for any of the synchronous transmitters or receivers to be enabled. NXP recommends that the source of the bit clock and frame sync is the last enabled and the first disabled.

When operating in Synchronous mode, only the bit clock, frame sync, transmitter enable, and receiver enable are shared. Otherwise, the separate SAI modules operate independently, although configuration register settings must be consistent across both the transmitter and receiver.

67.3.3 Frame sync configuration

When enabled, SAI continuously transmits and receives frames of data. Each frame consists of a fixed number of words, with each word consisting of a fixed number of bits. Within each frame, you can mask any word, causing the receiver to ignore that word and the transmitter to 3-state during that word.

The frame sync signal indicates the start of a frame. A valid frame sync requires the detection of a rising edge (if active high) or falling edge (if active low). The transmitter or receiver cannot be busy with a previous frame. SAI ignores a valid frame sync (in Target mode) or does not generate one (in Controller mode) for the first four bit-clock cycles after enabling the transmitter or receiver.

You can configure the transmitter and receiver frame sync independently by using any of the following options:

- Generate externally or internally
- Configure to be active high or active low
- Assert with the first bit in the frame or one bit early
- Assert for a duration between one bit-clock cycle and the first word length
- Configure the frame length from 1 to 32 words
- Configure the word length from 8 to 32 bits, with separate configurations for the length of the first word and that of the remaining words
- Transmit or receive words MSB first or LSB first

You cannot change these configuration options after enabling the SAI transmitter or receiver.

67.3.4 Data FIFO

Each transmit and receive channel includes a 8 × 32-bit FIFO. Use Transmit Data (TDR0 - TDR1) and Receive Data (RDR0 - RDR1) to access FIFO data.

67.3.4.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit-wide register via the First Bit Shifted (FBT) configuration field. This field selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required FBT configuration are shown in Figure 391 for LSB-first configurations and Figure 392 for MSB-first configurations.

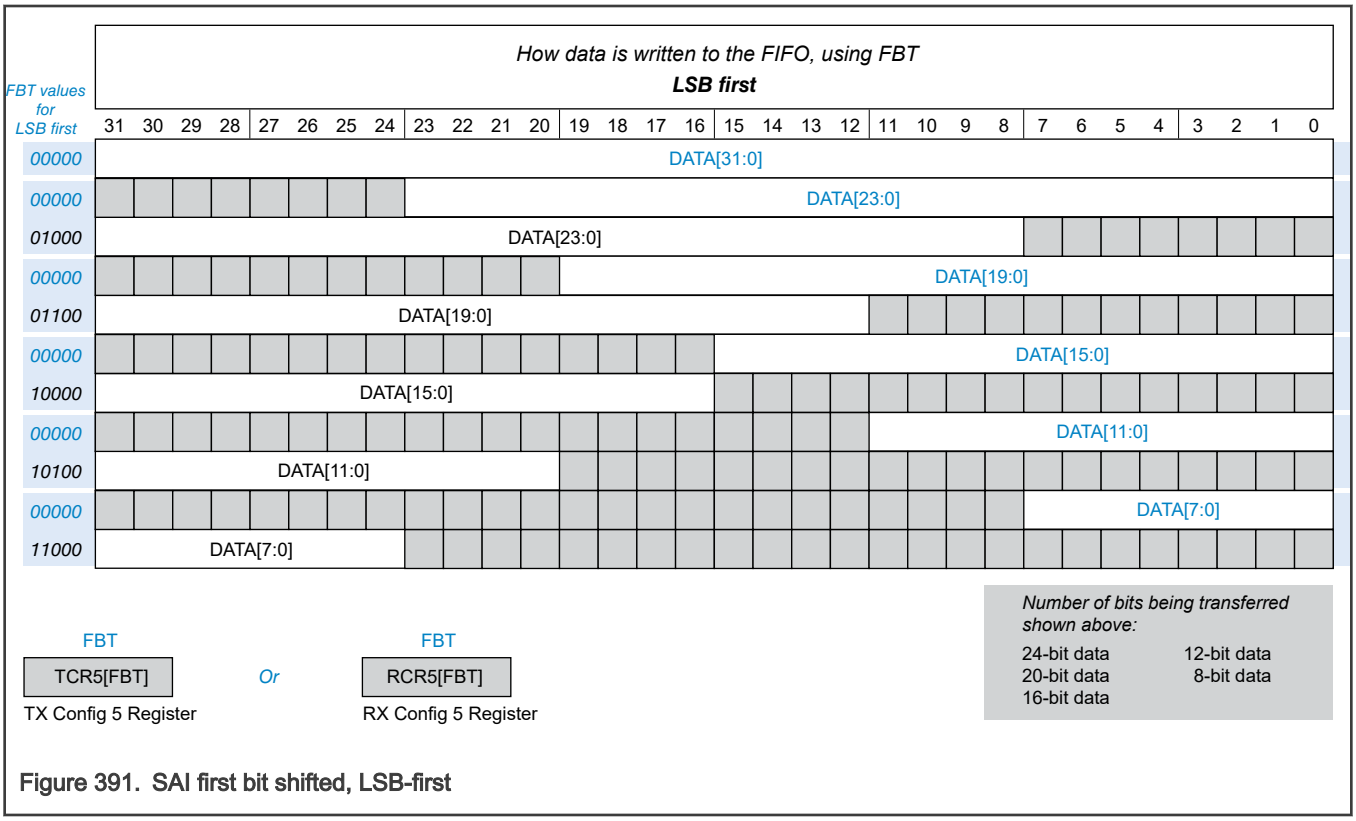


Figure 391. SAI first bit shifted, LSB-first

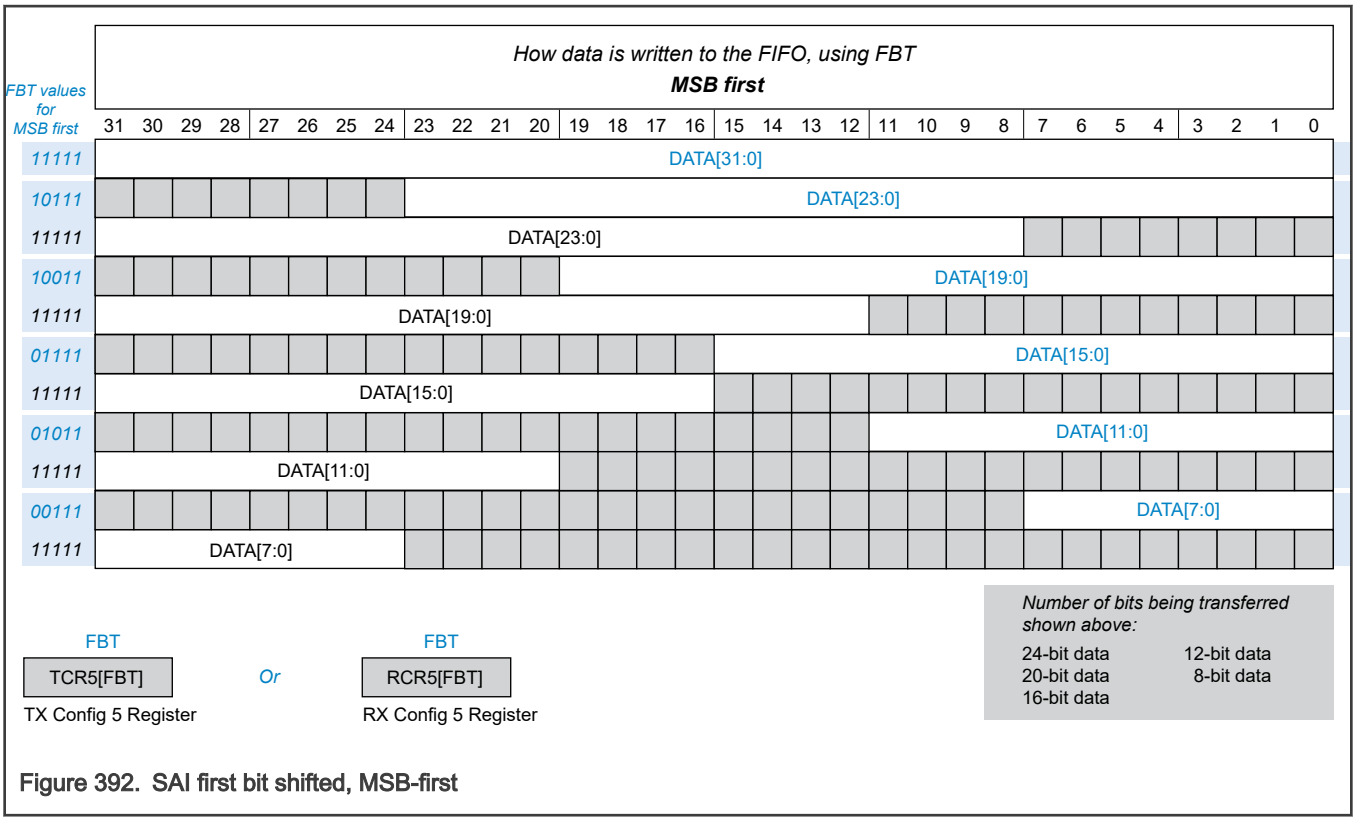


Figure 392. SAI first bit shifted, MSB-first

67.3.4.2 FIFO pointers

When writing to one of the Transmit Data (TDR0 - TDR1) registers, the write FIFO pointer of the corresponding Transmit FIFO register (TFR*n*[WFP]) increments after each valid write. SAI supports 8-bit, 16-bit, and 32-bit writes to TDR*n*, and the FIFO pointer increments after each individual write. Use only 8-bit writes when transmitting up to 8-bit data, and use only 16-bit writes when transmitting up to 16-bit data:

- If the transmit FIFO is full, SAI ignores writes to TDR*n*.
- If the transmit FIFO is empty, write to TDR*n* at least three bit-clock cycles before the start of the next unmasked word. This write avoids a FIFO underrun.
- Before enabling the transmitter, initialize the transmit FIFO with data. The transmitter starts a new frame after enabling the transmitter. If no data is in the FIFO, the transmitter immediately generates an error.

When reading one of the Receive Data (RDR0 - RDR1) registers, the read FIFO pointer of the corresponding Receive FIFO register (RFR*n*[RFP]) increments after each valid read. SAI supports 8-bit, 16-bit, and 32-bit reads from RDR*n*, and the FIFO pointer increments after each individual read. Use only 8-bit reads when receiving up to 8-bit data, and use only 16-bit reads when receiving up to 16-bit data:

- If the receive FIFO is empty, SAI ignores reads to RDR*n*.
- If the receive FIFO is full, read RDR*n* at least three bit-clock cycles before the end of an unmasked word. This read avoids a FIFO overrun.

67.3.4.3 FIFO packing

Using FIFO packing, you can store multiple 8-bit or 16-bit data words in one 32-bit FIFO word for the transmitter or receiver. You could emulate this feature by adjusting the number of bits per word and number of words per frame. FIFO packing, however, does not require even multiples of words per frame, and it fully supports word masking.

When FIFO packing is enabled, the FIFO pointers only increment when the full 32-bit FIFO word has been written (transmit) or read (receive). In this way, FIFO packing supports scenarios where different words within each frame are stored in different areas of memory.

Using 16-bit FIFO packing for transmitting, the transmit shift register loads at the start of each frame and after every second unmasked transmit word. The first transmitted word is taken from the 16-bit word at byte offset 0h. (The first bit is selected by TCR5[FBT], and you must configure it within this 16-bit word.) The second transmitted word is taken from the 16-bit word at byte offset 2h. (The first bit is selected by TCR5[FBT][3:0].) After the 16-bit word has been transmitted, the transmitter transmits logic zeroes until the start of the next word.

Using 16-bit FIFO packing for receiving, the receive shift register is stored after every second unmasked received word. If there is an odd number of unmasked-received words in each frame, the receive register is also stored, at the end of each frame. The first received word is stored in the 16-bit word at byte offset 0h. (The first bit is selected by RCR5[FBT], and you must configure it within this 16-bit word.) The second received word is stored in the 16-bit word at byte offset 2h. (The first bit is selected by RCR5[FBT][3:0].) The receiver ignores received data until the start of the next word after the 16-bit word has been received.

8-bit FIFO packing is similar to 16-bit packing, except four words are loaded or stored into each 32-bit FIFO word. The first word is stored in byte offset 0h, the second word in byte offset 1h, and so on. You must configure TCR5[FBT], RCR5[FBT], or both within byte offset 0h.

67.3.4.4 FIFO Combining mode

This mode enables separate FIFOs for multiple data channels to be used as a single FIFO for software accesses, as a single data channel, or both. The enabled data channels must be contiguous and data channel 0 must be enabled when using FIFO Combining mode.

You can combine FIFOs for software access by writing to transmit FIFO registers and reading from receive FIFO registers. Doing so enables a DMA controller or software to read or write multiple FIFOs without incrementing the address that is accessed. After FIFO Combining mode is enabled (TCR4[FCOMB] > 00b), the first software access to a FIFO register accesses the first enabled channel FIFO. The second access to a FIFO register accesses the second enabled channel FIFO. This process continues until software accesses the last enabled channel FIFO and the pointer resets to the first enabled channel FIFO. To reset the pointer manually, you can reset the FIFOs or disable FIFO combining on software accesses.

Combining FIFOs for transmit data channels enables one data channel to use the FIFOs of all enabled channel FIFOs. In this case, the data output is identical on each enabled data channel. The transmit shift registers for all enabled data channels load at the start of each frame. The registers also load upon every *n*th unmasked word, where *n* is the number of enabled data channels. The first word transmitted loads from the first enabled channel FIFO; the second word transmitted loads from the second enabled channel FIFO. This loading continues until the end of the frame.

Combining FIFOs for receive data channels enables one data channel to use the FIFOs of all enabled channel FIFOs. In this case, the received data from channel 0 is stored in each enabled data channel. The receive shift registers for all enabled data channels are stored after every *n*th unmasked word, where *n* is the number of enabled data channels. The first word received is stored in the first enabled channel FIFO; the second word received is stored in the second enabled channel FIFO. This storage continues until the end of the frame.

NOTE

The first word in each frame is always stored in the first enabled data channel. NXP recommends that the number of unmasked words in each frame be evenly divisible by the number of enabled data channels.

Combining FIFOs for data channels loads or stores each channel FIFO at the same time. As a result, FIFO error conditions are only checked every *n*th word, where *n* is the number of enabled data channels. If any enabled data channel meets the warning flag or request flag conditions, SAI asserts the FIFO warning and request flags.

67.3.5 Word mask register

The SAI transmitter and receiver each have a word mask register (Transmit Mask (TMR) and Receive Mask (RMR)) that you can use to mask any word in the frame. The word mask register is double buffered. You can update it before the end of each frame to mask a particular word in the next frame.

TMR causes the transmit data pin to be 3-stated for the length of each selected word; the transmit FIFO is not read for masked words.

RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

67.3.6 Clocking

- Audio controller clock
- Bit clock
- Bus clock

67.3.6.1 Audio controller clock

The audio controller clock generates the bit clock when you configure the receiver or transmitter for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio controller clocks to generate the bit clock.

The SAI peripheral controls the pin direction and postdivider of an audio controller clock. The postdivider can divide down the clock output on the MCLK signal pin without affecting the audio controller clock that the transmitter and receiver use.

The audio controller clock generation and selection is chip-specific. See chip-specific clocking information about how the audio controller clocks are generated.

67.3.6.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that an audio controller clock can generate internally or an external source can supply. SAI can also have synchronous bit clock and frame sync operation between the receiver and transmitter, or between multiple SAI peripherals. The transmitter and receiver configuration affects the bit clock and frame sync in the following ways:

- If you configure both the transmitter and receiver for asynchronous operation, each uses its own bit clock and frame sync.
- If you configure the transmitter for asynchronous operation and the receiver for synchronous operation, both use the transmitter bit clock and frame sync.
- If you configure the receiver for asynchronous operation and the transmitter for synchronous operation, both use the receiver bit clock and frame sync.

Your choice of synchronous or asynchronous operation selects the bit clock and frame sync used.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled.
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames.

In asynchronous operation, a SAI transmitter or receiver can use a bit clock externally generated by a SAI module instance that is disabled in Stop mode. In this case, disable the transmitter or receiver before entering Stop mode. This issue does not apply when the transmitter or receiver is in synchronous operation because all synchronous SAI modules are enabled and disabled simultaneously.

67.3.6.3 Bus clock

The control and configuration registers use the bus clock to generate synchronous interrupts and DMA requests.

NOTE

Although no minimum bus clock frequency is specified, the frequency must be fast enough (relative to the bit clock frequency) to serve the FIFOs. You must meet this requirement without generating a transmitter FIFO underrun or receiver FIFO overflow condition.

67.3.7 Reset

SAI is asynchronously reset on system reset. SAI has a Software reset and a FIFO reset.

67.3.7.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. The SAI receiver includes a software reset that resets all receiver internal logic, including bit clock generation, status flags, and FIFO pointers.

These software resets do not reset the configuration registers. The software resets remain asserted until you clear them via software.

67.3.7.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the value of the FIFO read pointer. This FIFO reset empties the FIFO contents. Use this reset after TCSR[FEF] becomes 1, and before SAI reinitializes the FIFO and TCSR[FEF] becomes 0. SAI asserts the FIFO reset for one cycle only.

The SAI transmitter can also reset the FIFO of individual data channels by writing 1 to the appropriate bit in TCR3[CFR]. Use this reset only when the corresponding bit in TCR3[TCE] is 0.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the value of the FIFO write pointer. This FIFO reset empties the FIFO contents. Use this reset after RCSR[FEF] becomes 1 and SAI reads any remaining data from the FIFO, and before RCSR[FEF] becomes 0. SAI asserts the FIFO reset for one cycle only.

The SAI receiver can also reset the FIFO of individual data channels by writing 1 to the appropriate bit in RCR3[RCE]. Use this reset only when the corresponding bit in RCR3[RCE] is 0.

67.3.8 Interrupts and DMA requests

In SAI, the transmitter and receiver generate separate interrupts and DMA requests, but use the same status flags. SAI generates asynchronous versions of the transmitter and receiver interrupts to wake the CPU from Stop mode. SAI only generates asynchronous interrupts when the system clock is gated but the corresponding BCLK signal is active.

67.3.8.1 FIFO request flag

SAI sets the transmit FIFO request flag (TCSR[FRF]) when the number of entries in any enabled transmit FIFO is less than or equal to the transmit FIFO watermark configuration (TCR1[TFW]). SAI clears this flag when the number of entries in each enabled transmit FIFO is greater than TCR1[TFW].

SAI sets the receive FIFO request flag (RCSR[FRF]) when the number of entries in any enabled receive FIFO is greater than the receive FIFO watermark configuration (RCR1[RFW]). SAI clears this flag when the number of entries in each enabled receive FIFO is less than or equal to RCR1[RFW].

The FIFO request flag can generate an interrupt when TCSR[FRIE] = 1. It can generate a DMA request when TCSR[FRDE] = 1.

67.3.8.2 FIFO warning flag

SAI sets the transmit FIFO warning flag (TCSR[FWF]) when the number of entries in any of the enabled transmit FIFOs is empty. SAI clears this flag when the number of entries in each enabled transmit FIFO is not empty.

SAI sets the receive FIFO warning flag (RCSR[FWF]) when the number of entries in any of the enabled receive FIFOs is full. SAI clears this flag when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an interrupt when TCSR[FWIE] = 1. The flag can generate a DMA request when TCSR[FWDE] = 1.

67.3.8.3 FIFO error flag

SAI sets the transmit FIFO error flag (TCSR[FEF]) when any enabled transmit FIFOs underrun. After the error flag is set, all enabled transmit channels transmit zero data before SAI clears TCSR[FEF].

When you write 1 to TCR4[FCONT], the FIFO continues transmitting data following an underrun without software intervention. To ensure that data transmits in the correct order, the transmitter continues from the same word number in the frame that caused the FIFO to underrun. It continues only after new data is written to the transmit FIFO. In this case, clear TCSR[FEF] without reinitializing the transmit FIFOs.

SAI sets the receive FIFO error flag (RCSR[FEF]) when any enabled receive FIFO overflows. After the flag is set, all enabled receive channels discard their received data until SAI clears RCSR[FEF] and the next receive frame starts. Empty all enabled receive FIFOs before SAI clears RCSR[FEF].

When you write 1 to RCR4[FCONT], the FIFO continues receiving data following an overflow without software intervention. To ensure that data is received in the correct order, the receiver continues from the same word number in the frame that caused the FIFO to overflow. It continues only after data has been read from the receive FIFO. In this case, clear RCSR[FEF] without emptying the receive FIFOs.

TCSR[FEF] and RSCR[FEF] can only generate an interrupt.

67.3.8.4 Sync error flag

SAI sets a sync error flag (TCSR[SEF] or RCSR[SEF]) when both of these conditions are true:

- The frame sync is generated externally.
- The external frame sync asserts when the transmitter or receiver is busy with the previous frame.

SAI ignores the external frame sync assertion and sets the sync error flag. When this flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

TCSR[SEF] and RCSR[SEF] can only generate an interrupt.

67.3.8.5 Word start flag

SAI sets the word start flag (TCSR[WSF]) at the start of the second bit-clock cycle for the selected word. You can select this word via TCR3[WDFL].

TCSR[WSF] can only generate an interrupt.

67.4 External signals

Name	Function	I/O
TX_BCLK	Transmit bit clock: the bit clock is an input when externally generated and an output when internally generated.	I/O
TX_SYNC	Transmit frame sync: the frame sync is an input sampled synchronously by the bit clock when externally generated. It is an output generated synchronously by the bit clock when internally generated.	I/O
TX_DATA[1:0]	Transmit data: the bit clock synchronously generates the transmit data; this signal is 3-stated when not transmitting a word.	O
RX_BCLK	Receive bit clock: the bit clock is an input when externally generated and an output when internally generated.	I/O

Table continues on the next page...

Table continued from the previous page...

Name	Function	I/O
RX_SYNC	Receive frame sync: the frame sync is an input sampled synchronously by the bit clock when externally generated. It is an output generated synchronously by the bit clock when internally generated.	I/O
RX_DATA[1:0]	Receive data: the bit clock synchronously samples the receive data.	I
MCLK	Audio controller clock: the audio controller clock is an input when externally generated and an output when internally generated.	I/O

67.5 Initialization

To initialize SAI:

1. Enable the clock to SAI.
2. Reset the internal transmitter logic and receiver logic by writing 1 to TCSR[SR] and RCSR[SR], respectively.

67.6 Memory map and register definition

A read or write access to an address from offset 108h and above results in a bus error.

67.6.1 SAI register descriptions

67.6.1.1 SAI memory map

SAI0 base address: 4010_6000h

SAI1 base address: 4010_7000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0302_0000h
4h	Parameter (PARAM)	32	R	0005_0302h
8h	Transmit Control (TCSR)	32	RW	0000_0000h
Ch	Transmit Configuration 1 (TCR1)	32	RW	0000_0000h
10h	Transmit Configuration 2 (TCR2)	32	RW	0000_0000h
14h	Transmit Configuration 3 (TCR3)	32	RW	0000_0000h
18h	Transmit Configuration 4 (TCR4)	32	RW	0000_0000h
1Ch	Transmit Configuration 5 (TCR5)	32	RW	0000_0000h
20h - 24h	Transmit Data (TDR0 - TDR1)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
40h - 44h	Transmit FIFO (TFR0 - TFR1)	32	R	0000_0000h
60h	Transmit Mask (TMR)	32	RW	0000_0000h
88h	Receive Control (RCSR)	32	RW	0000_0000h
8Ch	Receive Configuration 1 (RCR1)	32	RW	0000_0000h
90h	Receive Configuration 2 (RCR2)	32	RW	0000_0000h
94h	Receive Configuration 3 (RCR3)	32	RW	0000_0000h
98h	Receive Configuration 4 (RCR4)	32	RW	0000_0000h
9Ch	Receive Configuration 5 (RCR5)	32	RW	0000_0000h
A0h - A4h	Receive Data (RDR0 - RDR1)	32	R	0000_0000h
C0h - C4h	Receive FIFO (RFR0 - RFR1)	32	R	0000_0000h
E0h	Receive Mask (RMR)	32	RW	0000_0000h
100h	MCLK Control (MCR)	32	RW	0000_0000h

67.6.1.2 Version ID (VERID)

Offset

Register	Offset
VERID	0h

Function

Contains version numbers for the module design and feature set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 MAJOR	Major Version Number Indicates the major version number for the specification.
23-16 MINOR	Minor Version Number Indicates the minor version number for the specification.
15-0 FEATURE	Feature Specification Number Indicates the feature set number. 0000_0000_0000_0000b - Standard feature set

67.6.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function
Contains parameter values implemented in the module.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												FRAME			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				FIFO				0				DATALINE			
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0

Fields

Field	Function
31-20 —	Reserved
19-16	Frame Size

Table continues on the next page...

Table continued from the previous page...

Field	Function
FRAME	Describes the maximum number of slots per frame, which is 2^FRAME.
15-12 —	Reserved
11-8 FIFO	FIFO Size Describes the number of words in each FIFO, which is 2^FIFO.
7-4 —	Reserved
3-0 DATALINE	Number of Data Lines Contains the number of data lines implemented.

67.6.1.4 Transmit Control (TCSR)

Offset

Register	Offset
TCSR	8h

Function

Contains transmitter enable fields including resets, error and interrupt enable fields, and error flag fields.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TE	STOPE	DBGE	BCE	0		0	SR	0			WSF	SEF	FEF	FWF	FRF
W							FR					W1C	W1C	W1C		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WSIE	SEIE	FEIE	FWIE	FRIE	0			0			FWDE	FRDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Transmitter Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TE	Enables the transmitter. When you write 0 to this field, the transmitter remains enabled, and this field remains 1 until the end of the current frame. 0b - Disable 1b - Enable (or transmitter has been disabled and has not yet reached the end of the frame)
30 STOPE	Stop Enable Enables transmitter operation in Stop mode. 0b - Disable 1b - Enable
29 DBGE	Debug Enable Enables transmitter operation in Debug mode, which does not affect the transmit bit clock. If you write 0 to this field, the transmitter operation is disabled after completing the current frame. 0b - Disable 1b - Enable
28 BCE	Bit Clock Enable Enables the transmit bit clock, separately from TCSR[TE]. This field automatically becomes 1 when TCSR[TE] becomes 1. When you write 0 to this field, the transmit bit clock remains enabled, and the field remains 1 until the end of the current frame. 0b - Disable 1b - Enable
27-26 —	Reserved
25 FR	FIFO Reset Empties the FIFO and sets the FIFO read and write pointers to the same value, which may or may not be zero. **NOTE** The FIFO reset is asserted for one cycle only. Reading this field always returns zero. SAI resets the FIFO pointers when the transmitter is disabled or the FIFO error flag is set. 0b - No effect 1b - FIFO reset
24 SR	Software Reset Resets the internal transmitter logic, including the FIFO read and write pointers. Software-visible registers are not affected, except for the status registers.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The software reset remains asserted until software writes 0 to the field. 0b - No effect 1b - Software reset
23-21 —	Reserved
20 WSF	Word Start Flag Indicates whether the start of the configured word has been detected. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
19 SEF	Sync Error Flag Indicates whether an error in the externally generated frame sync has been detected. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
18 FEF	FIFO Error Flag Indicates whether an enabled transmit FIFO has underrun. **NOTE** This field behaves differently for register reads and writes. When reading

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
17 FWF	FIFO Warning Flag Indicates whether an enabled transmit FIFO is empty. 0b - Not empty 1b - Empty
16 FRF	FIFO Request Flag Indicates whether the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark. 0b - Watermark not reached 1b - Watermark reached
15-13 —	Reserved
12 WSIE	Word Start Interrupt Enable Enables word start interrupts. 0b - Disable 1b - Enable
11 SEIE	Sync Error Interrupt Enable Enables sync error interrupts. 0b - Disable 1b - Enable
10 FEIE	FIFO Error Interrupt Enable Enables FIFO error interrupts. 0b - Disable 1b - Enable
9 FWIE	FIFO Warning Interrupt Enable Enables FIFO warning interrupts. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
8 FRIE	FIFO Request Interrupt Enable Enables FIFO request interrupts. 0b - Disable 1b - Enable
7-5 —	Reserved
4-2 —	Reserved
1 FWDE	FIFO Warning DMA Enable Enables the DMA warning. 0b - Disable 1b - Enable
0 FRDE	FIFO Request DMA Enable Enables DMA requests. 0b - Disable 1b - Enable

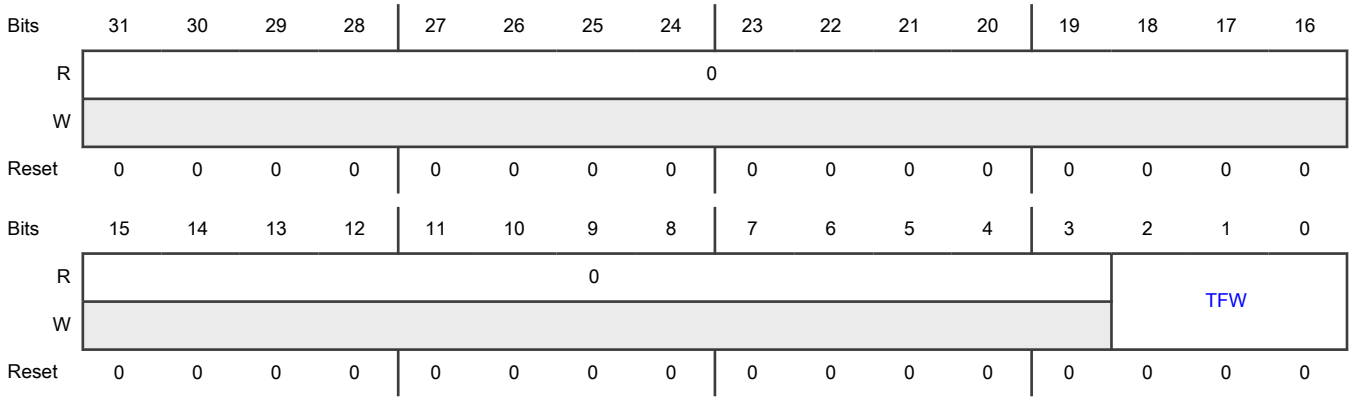
67.6.1.5 Transmit Configuration 1 (TCR1)

Offset

Register	Offset
TCR1	Ch

Function
Configures the watermark level for all enabled transmit channels.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 TFW	Transmit FIFO Watermark Indicates the number of 32-bit FIFO words in the watermark level of the transmit FIFO. 000b - 1 001b - 2 010b-110b - (TFW +1) 111b - 8

67.6.1.6 Transmit Configuration 2 (TCR2)

Offset

Register	Offset
TCR2	10h

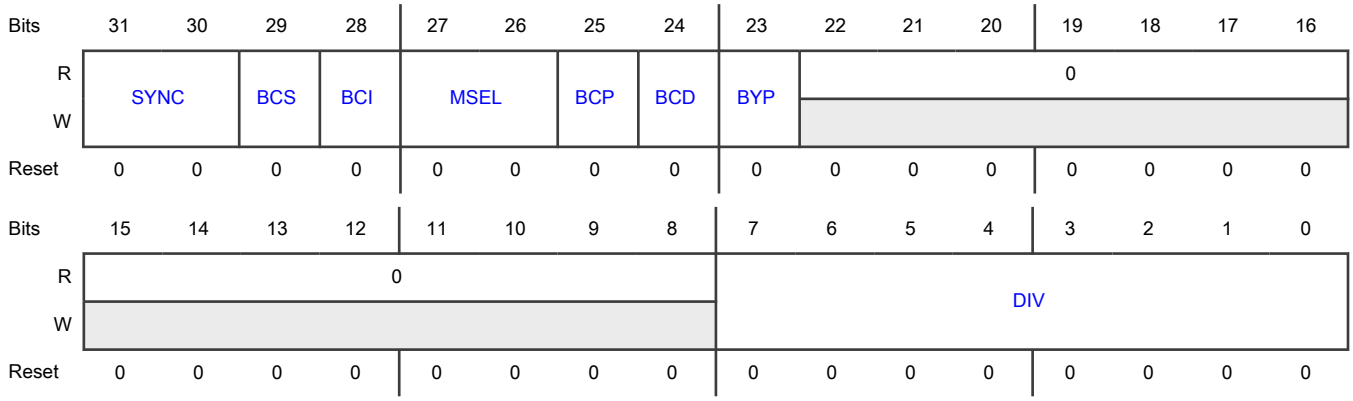
Function

Contains the SYNC mode and clock setting fields.

NOTE

Do not alter this register when TCSR[TE] is 1.

Diagram



Fields

Field	Function
31-30 SYNC	Synchronous Mode Configures Asynchronous and Synchronous modes of operation. When configured for Synchronous mode, you must configure the receiver or other SAI peripheral for asynchronous operations. 00b - Asynchronous mode 01b - Synchronous with receiver 10b - Synchronous with another SAI transmitter 11b - Synchronous with another SAI receiver
29 BCS	Bit Clock Swap Swaps the bit clock used by the transmitter. When you configure the transmitter in Asynchronous mode and this field is 1, the receiver bit clock (RX_BCLK) clocks the transmitter. This configuration allows the transmitter and receiver to share a bit clock, while the transmitter continues to use the transmit frame sync (TX_SYNC). When you configure the transmitter in Synchronous mode, you must write the same value to the transmitter BCS field and the receiver BCS field. When both are 1, the transmitter and receiver are both clocked by the transmitter bit clock (TX_BCLK) but both use the receiver frame sync (RX_SYNC). This field has no effect when synchronous to another SAI peripheral. 0b - Use the normal bit clock source 1b - Swap the bit clock source
28 BCI	Bit Clock Input Enables the internal logic to be clocked as if the bit clock is generated externally. When this field is 1 while using an internally generated bit clock in Synchronous or Asynchronous mode, the bit clock used by the transmitter is delayed by the pad output delay. (The pad input clocks the transmitter as if the clock is externally generated.) This setting decreases the data input setup time, but increases the data output valid time.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Use the Target mode timing from the data sheet for the transmitter when this field is 1. In Synchronous mode, this field allows the transmitter to use the Target mode timing from the data sheet, while the receiver uses the Controller mode timing. This field has no effect when configured for an externally generated bit clock or when synchronous to another SAI peripheral. **NOTE** When BCI = 1, both the input buffer and output buffer must be enabled for the BCLK pad. 0b - Disable 1b - Enable
27-26 MSEL	MCLK Select Selects the audio controller clock option used to generate an internally generated bit clock. This field has no effect when SAI is configured for an externally generated bit clock. **NOTE** Depending on the chip, some controller clock options might not be available. See the chip-specific information for the meaning of each option. **NOTE** Choose the same value for this field and for RCR2[MSEL]. 00b - Bus clock 01b - Controller clock (MCLK) option 1 10b - Controller clock (MCLK) option 2 11b - Controller clock (MCLK) option 3
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. If you write 0 to this field, the bit clock becomes active high with drive outputs on rising edge and sample inputs on falling edge. If you write 1 to this field, the bit clock becomes active low with drive outputs on falling edge and sample inputs on rising edge. 0b - Active high 1b - Active low
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0b - Generate externally in Target mode 1b - Generate internally in Controller mode
23 BYP	Bit Clock Bypass Bypasses the bit clock divider. When bypassed, the internal bit clock is divide-by-one of the audio controller clock. When not bypassed, the internal bit clock is generated from the bit clock divider.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE Choose the same value for this field and for RCR2[BYP]. 0b - Disable 1b - Enable
22-8 —	Reserved
7-0 DIV	Bit Clock Divide Determines the value by which SAI divides down the audio controller clock to generate the bit clock (when configured for an internal bit clock). The division value is (DIV + 1) × 2.

67.6.1.7 Transmit Configuration 3 (TCR3)

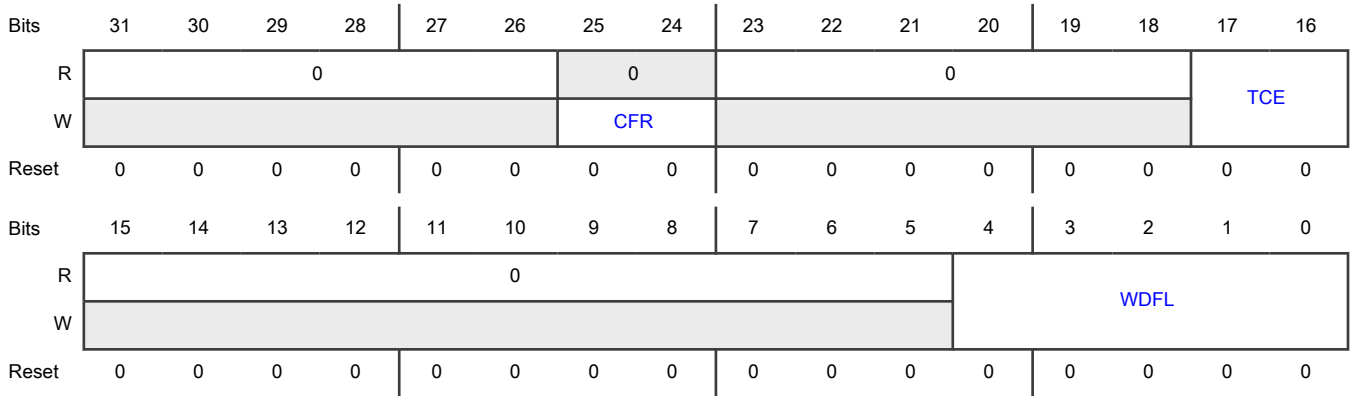
Offset

Register	Offset
TCR3	14h

Function

Contains the transmit channel settings.

Diagram



Fields

Field	Function
31-26	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
25-24 CFR	Channel FIFO Reset Resets the FIFO pointers for a specific channel. Reading this field always returns zero. You must reset FIFO pointers only when a channel is disabled or the FIFO error flag is set. The width of this field = the number of transmit channels (call it *n*). For example, if this field is 2 bits wide, bit position 24 refers to the transmit channel 1 FIFO pointer and bit position 25 refers to the transmit channel 2 FIFO pointer. Writing 1 to bit 24 resets the transmit channel 1 FIFO pointer, and writing 1 to bit 25 enables the transmit channel 2 FIFO pointer. Writing 1 to bit *n* resets the transmit channel *n* FIFO pointer. **NOTE** When there is only one channel, you do not need the individual channel FIFO reset (TCR3[CFR]). For a single channel, use the global FIFO reset (TCSR[FR]). • 0b – No effect • 1b – Reset transmit data channel *n* FIFO
23-18 —	Reserved
17-16 TCE	Transmit Channel Enable Enables the corresponding data channel for transmit operations. Changing this field takes effect immediately for generating the FIFO request and warning flags. It takes effect at the end of each frame for transmit operations. The width of this field = the number of transmit channels (call it *n*). For example, if this field is two bits wide, bit position 16 refers to transmit channel 1 and bit position 17 refers to transmit channel 2. Writing 1 to bit 16 enables transmit channel 1, and setting bit 17 enables transmit channel 2. Writing 1 to bit *n* enables transmit channel *n*. For each transmit data channel: • 0b – Disable • 1b – Enable
15-5 —	Reserved
4-0 WDFL	Word Flag Configuration Selects the word that sets the word start flag (TCSR[WSF]). The value must be one less than the word number. For example, writing 0 selects the first word in the frame. When you configure this field with a value greater than TCR4[FRSZ], TCSR[WSF] is never set.

67.6.1.8 Transmit Configuration 4 (TCR4)

Offset

Register	Offset
TCR4	18h

Function

Contains the transmit fields for FIFO Combine mode, FIFO Packing mode, and frame sync settings.

NOTE

Do not alter this register when TCSR[TE] is 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			FCONT	FCOMB		FPACK		0				FRSZ			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			SYWD					0		CHMOD	MF	FSE	ONDEM	FSP	FSD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 FCONT	FIFO Continue on Error Configures when SAI must continue transmitting after a FIFO error is detected. When this field is 0 and a FIFO error occurs, SAI continues from the start of the next frame after the FIFO error flag clears. When this field is 1 and a FIFO error occurs, SAI continues from the same word that caused the FIFO error after the FIFO warning flag clears. 0b - Continue from the start of the next frame 1b - Continue from the same word that caused the FIFO error
27-26 FCOMB	FIFO Combine Mode Enables FIFO Combine mode for specified operations. When FIFO Combine mode is enabled for FIFO writes, software writing to any FIFO data register alternates the write among the enabled data channel FIFOs. For example, if two data channels are enabled:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• The first write is performed to the first enabled data channel FIFO. • The second write is performed to the second enabled data channel FIFO. Resetting the FIFO or disabling FIFO Combine mode for FIFO writes resets the pointer back to the first enabled data channel. When FIFO Combine mode is enabled for FIFO reads from the transmit shift registers, the transmit data channel output alternates between the enabled data channel FIFOs. For example, if two data channels are enabled: • The first unmasked word is transmitted from the first enabled data channel FIFO. • The second unmasked word is transmitted from the second enabled data channel FIFO. Because the first word of the frame is always transmitted from the first enabled data channel FIFO, NXP recommends that the number of unmasked words per frame is evenly divisible by the number of enabled data channels. 00b - Disable 01b - Enable on FIFO reads (from transmit shift registers) 10b - Enable on FIFO writes (by software) 11b - Enable on FIFO reads (from transmit shift registers) and writes (by software)
25-24 FPACK	FIFO Packing Mode Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8 bits or 16 bits, only the first 8 bits or 16 bits are loaded from the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO write pointer only increments when you write the full 32-bit FIFO word. 00b - Disable FIFO packing 01b - Reserved 10b - Enable 8-bit FIFO packing 11b - Enable 16-bit FIFO packing
23-21 —	Reserved
20-16 FRSZ	Frame Size Configures the number of words in each frame. The value must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15-13 —	Reserved
12-8 SYWD	Sync Width Configures the length of the frame sync in number of bit-clock cycles. The value must be one less than the number of bit-clock cycles. For example, write 0 for the frame sync to assert for one bit-clock cycle only. You cannot configure the sync width to be longer than the first word of the frame.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-6 —	Reserved
5 CHMOD	Channel Mode Specifies the mode of transmit data pins. In TDM mode, transmit data pins are 3-stated when slots are masked or channels are disabled. In Output mode, transmit data pins are never 3-stated, and they output zero when slots are masked or channels are disabled. 0b - TDM mode 1b - Output mode
4 MF	MSB First Configures what is transmitted first: LSB or MSB. 0b - LSB 1b - MSB
3 FSE	Frame Sync Early Determines when frame sync is asserted. 0b - First bit of the frame 1b - One bit before the first bit of the frame
2 ONDEM	On-Demand Mode Determines when the internal frame sync is generated. When this field is 1, and the frame sync is generated internally, a frame sync is only generated after the FIFO warning flag is cleared. 0b - Generated continuously 1b - Generated after the FIFO warning flag is cleared
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0b - Active high 1b - Active low
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0b - Generated externally in Target mode 1b - Generated internally in Controller mode

67.6.1.9 Transmit Configuration 5 (TCR5)

Offset

Register	Offset
TCR5	1Ch

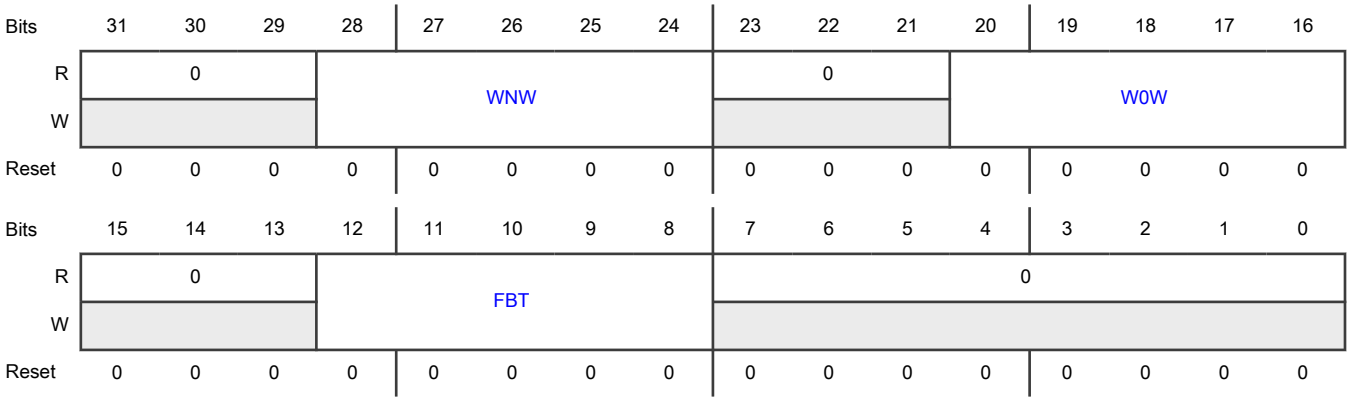
Function

Contains transmit word width and bit index settings.

NOTE

Do not alter this register when TCSR[TE] is 1.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-24 WNW	Word N Width Configures the number of bits in each word, except for the first one in the frame. The value written must be one less than the number of bits per word. SAI does not support word widths of less than 8 bits. SAI does not support bit settings 0–6 (0b–0_0110b). 0_0111b - 8 0_1000b - 9 0_1001b-1_1110b - (WNW value + 1) 1_1111b - 32
23-21 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
20-16 W0W	Word 0 Width Configures the number of bits in the first word of each frame. The value written must be one less than the number of bits in the first word. SAI does not support word widths of less than 8 bits when there is only one word per frame. SAI does not support bit settings 0–6 (0b–0_0110b). 0_0111b - 8 0_1000b - 9 0_1001b-1_1110b - (W0W value + 1) 1_1111b - 32
15-13 —	Reserved
12-8 FBT	First Bit Shifted Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB-first, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB-first, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB-first. The value written must be less than or equal to 31-word width when configured for LSB-first. See Data alignment for details. 0_0000b - 0 0_0001b-1_1110b - FBT 1_1111b - 31
7-0 —	Reserved

67.6.1.10 Transmit Data (TDR0 - TDR1)

Offset

Register	Offset
TDR0	20h
TDR1	24h

Function

Contains transmit data.

When the transmit FIFO is not full, writes to this register push the data written into the transmit data FIFO. When the transmit FIFO is full, SAI ignores writes to this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	TDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	TDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 TDR	Transmit Data

67.6.1.11 Transmit FIFO (TFR0 - TFR1)

Offset

Register	Offset
TFR0	40h
TFR1	44h

Function

Contains the transmit FIFO pointers. The MSB of the read and write pointers distinguishes between FIFO full and empty conditions. If the read and write pointers are identical, the FIFO is empty. If the read and write pointers are identical except for the MSB, the FIFO is full.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WCP	0											WFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												RFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 WCP	Write Channel Pointer Indicates whether this data channel is the next FIFO to be written when FIFO Combine mode is enabled for write operations. 0b - No effect 1b - Next FIFO to be written
30-20 —	Reserved
19-16 WFP	Write FIFO Pointer Indicates the FIFO write pointer for the transmit data channel.
15-4 —	Reserved
3-0 RFP	Read FIFO Pointer Indicates the FIFO read pointer for the transmit data channel.

67.6.1.12 Transmit Mask (TMR)

Offset

Register	Offset
TMR	60h

Function

Contains the mask for the transmit word. This register is double-buffered and updates:

- When TCSR[TE] first becomes 1.
- At the end of each frame.

This setup allows the masked words in each frame to change from frame to frame.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TWM															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TWM															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 TWM	Transmit Word Mask Configures whether the transmit word *n* is masked for the corresponding word in the frame. In this case, being masked means that transmit data pins are 3-stated or driven zero and transmit data is not read from the FIFO. 0000_0000_0000_0000_0000_0000_0000_0000b - Enable 0000_0000_0000_0000_0000_0000_0000_0001b - Mask

67.6.1.13 Receive Control (RCSR)

Offset

Register	Offset
RCSR	88h

Function

Contains receiver enable fields including resets, error and interrupt enable fields, and error flag fields.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RE	STOPE	DBGE	BCE	0		0	SR	0			WSF	SEF	FEF	FWF	FRF
W							FR					W1C	W1C	W1C		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			WSIE	SEIE	FEIE	FWIE	FRIE	0			0			FWDE	FRDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 RE	Receiver Enable Enables the receiver. When you write 0 to this field, the receiver remains enabled and the field remains 1 until the end of the current frame. 0b - Disable 1b - Enable (or receiver disabled and not yet reached end of frame)
30 STOPE	Stop Enable Enables receiver operation in Stop mode. 0b - Disable 1b - Enable
29 DBGE	Debug Enable Enables receiver operation in Debug mode, which does not affect the receive bit clock. 0b - Disable after completing the current frame 1b - Enable
28 BCE	Bit Clock Enable Enables the receive bit clock, separately from RCSR[RE]. This field becomes 1 automatically when RCSR[RE] becomes 1. When you write 0 to this field, the receive bit clock remains enabled, and this field remains 1 until the end of the current frame. 0b - Disable 1b - Enable
27-26 —	Reserved
25 FR	FIFO Reset Empties the FIFO, and sets the FIFO read and write pointers to the same value, which can be zero. Reading this field always returns zero. **NOTE** The FIFO reset is asserted for one cycle only. You must reset FIFO pointers only when the receiver is disabled or the FIFO error flag is 1. 0b - No effect 1b - Reset
24 SR	Software Reset Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The software reset remains asserted until you clear it. 0b - No effect 1b - Software reset
23-21 —	Reserved
20 WSF	Word Start Flag Indicates whether SAI has detected the start of the configured word. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
19 SEF	Sync Error Flag Indicates whether SAI has detected an error in the externally generated frame sync. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
18 FEF	FIFO Error Flag Indicates whether an enabled receive FIFO has overflowed. **NOTE** This field behaves differently for register reads and writes. When reading

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No error 1b - Receive overflow detected When writing 0b - No effect 1b - Clear the flag
17 FWF	FIFO Warning Flag Indicates whether an enabled receive FIFO is full. 0b - Not full 1b - Full
16 FRF	FIFO Request Flag Indicates whether the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark. 0b - Watermark not reached 1b - Watermark reached
15-13 —	Reserved
12 WSIE	Word Start Interrupt Enable Enables word start interrupts. 0b - Disable 1b - Enable
11 SEIE	Sync Error Interrupt Enable Enables sync error interrupts. 0b - Disable 1b - Enable
10 FEIE	FIFO Error Interrupt Enable Enables FIFO error interrupts. 0b - Disable 1b - Enable
9 FWIE	FIFO Warning Interrupt Enable Enables FIFO warning interrupts. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
8 FRIE	FIFO Request Interrupt Enable Enables FIFO request interrupts. 0b - Disable 1b - Enable
7-5 —	Reserved
4-2 —	Reserved
1 FWDE	FIFO Warning DMA Enable Enables DMA warnings. 0b - Disable 1b - Enable
0 FRDE	FIFO Request DMA Enable Enables DMA requests. 0b - Disable 1b - Enable

67.6.1.14 Receive Configuration 1 (RCR1)

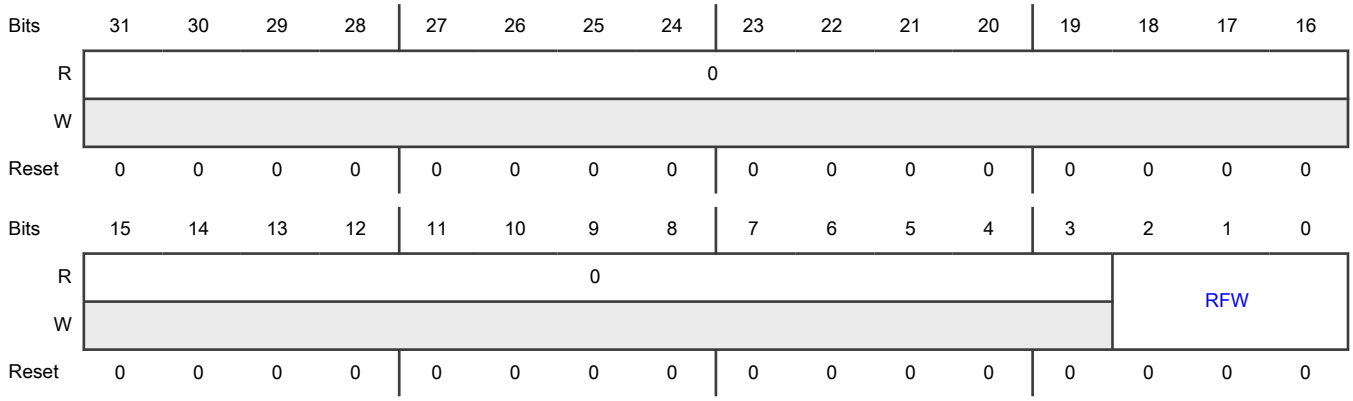
Offset

Register	Offset
RCR1	8Ch

Function

Configures the watermark level for all enabled receiver channels.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 RFW	Receive FIFO Watermark Configures the level of the receive FIFO watermark, in 32-bit FIFO words. 000b - 1 001b - 2 010b-110b - (RFW value + 1) 111b - 8

67.6.1.15 Receive Configuration 2 (RCR2)

Offset

Register	Offset
RCR2	90h

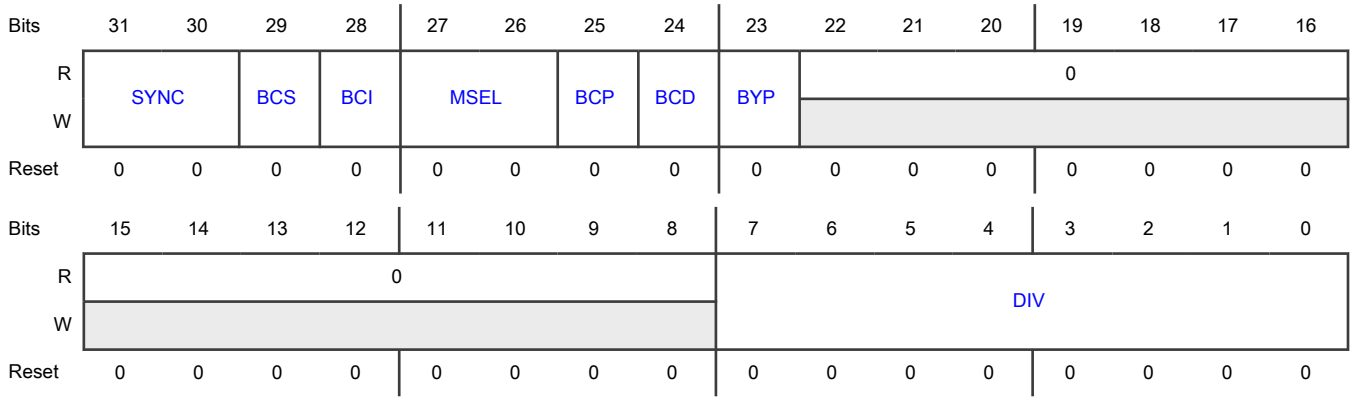
Function

Contains the SYNC mode and clock setting fields.

NOTE

Do not alter this register when RCSR[RE] is 1.

Diagram



Fields

Field	Function
31-30 SYNC	Synchronous Mode Configures Asynchronous and Synchronous modes of operation. When configured for a Synchronous mode of operation, you must configure the transmitter or other SAI peripheral for asynchronous operation. 00b - Asynchronous mode 01b - Synchronous with transmitter 10b - Synchronous with another SAI receiver 11b - Synchronous with another SAI transmitter
29 BCS	Bit Clock Swap Swaps the bit clock used by the receiver. When the receiver is configured in Asynchronous mode and this field is 1, the transmitter bit clock (TX_BCLK) clocks the receiver. This setting allows the transmitter and receiver to share one bit clock, while the receiver continues to use the receiver frame sync (RX_SYNC). When you configure the receiver in Synchronous mode, you must write the same value to the transmitter BCS field and receiver BCS field. When both are 1, the transmitter and receiver are both clocked by the receiver bit clock (RX_BCLK) but both use the transmitter frame sync (TX_SYNC). This field has no effect when synchronous to another SAI peripheral. 0b - Use the normal bit clock source 1b - Swap the bit clock source
28 BCI	Bit Clock Input Enables the internal logic to be clocked as if the bit clock is generated externally. When this field is 1 and when using an internally generated bit clock in Synchronous or Asynchronous mode, the bit clock used by the receiver is delayed by the pad output delay. (The pad input clocks the receiver as if the clock is externally generated.) This setting decreases the data input setup time, but increases the data output valid time.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Use the Target mode timing from the data sheet for the receiver when this field is 1. In Synchronous mode, this field allows the receiver to use the Target mode timing from the data sheet, while the transmitter uses the Controller mode timing. This field has no effect when configured for an externally generated bit clock or when synchronous to another SAI peripheral. **NOTE** When this field is 1, both the input buffer and output buffer must be enabled for the BCLK pad. 0b - Disable 1b - Enable
27-26 MSEL	MCLK Select Selects the audio controller clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE** Some controller clock options are not available for some chips. See the chip-specific information for the availability and chip-specific meaning of each option. **NOTE** Choose the same value for this field and for TCR2[MSEL]. 00b - Bus clock 01b - Controller clock (MCLK) option 1 10b - Controller clock (MCLK) option 2 11b - Controller clock (MCLK) option 3
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. If you write 0 to this field, the bit clock becomes active high with drive outputs on the rising edge and sample inputs on the falling edge. If you write 1 to this field, the bit clock becomes active low with drive outputs on the falling edge and sample inputs on the rising edge. 0b - Active high 1b - Active low
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0b - Generated externally in Target mode 1b - Generated internally in Controller mode
23 BYP	Bit Clock Bypass Enables the bypass of the bit clock divider. When enabled, the internal bit clock is divide-by-one of the audio controller clock. When disabled, the internal bit clock is generated from the bit clock divider.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE Choose the same value for this field and for TCR2[BYP]. 0b - Disable 1b - Enable
22-8 —	Reserved
7-0 DIV	Bit Clock Divide Determines the value by which the audio controller clock is divided to generate the bit clock, when configured for an internal bit clock. The division value is (DIV + 1) × 2.

67.6.1.16 Receive Configuration 3 (RCR3)

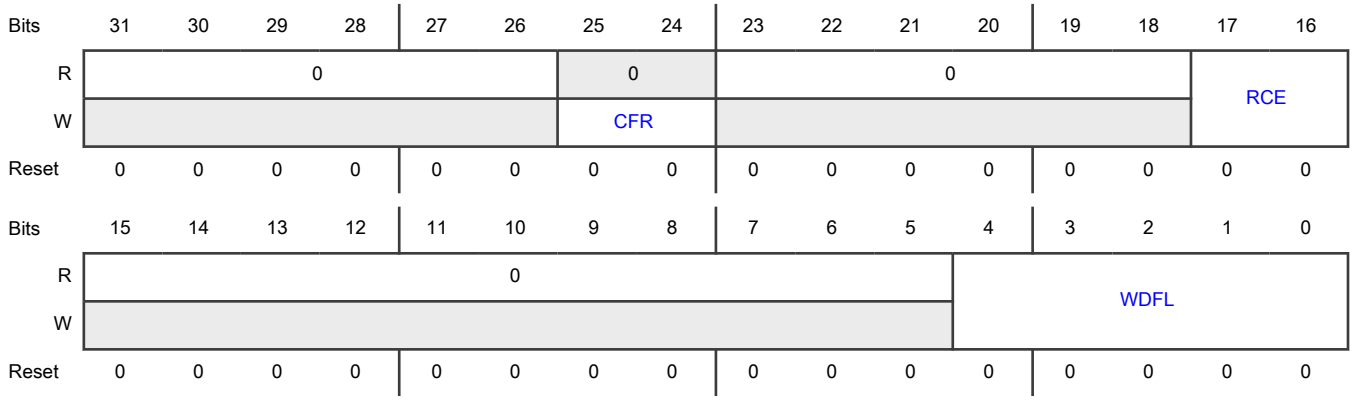
Offset

Register	Offset
RCR3	94h

Function

Contains the receive channel settings.

Diagram



Fields

Field	Function
31-26	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
25-24 CFR	Channel FIFO Reset Resets the FIFO pointers for a specific channel. Reading this field always returns zero. Reset FIFO pointers only when a channel is disabled or the FIFO error flag is set. The width of this field = the number of receive channels (call it *n*). For example, if this field is two bits wide, bit position 24 refers to the receive channel 1 FIFO pointer and bit position 25 refers to the receive channel 2 FIFO pointer. Writing 1 to bit 24 resets the receive channel 1 FIFO pointer, and writing 1 to bit 25 enables the receive channel 2 FIFO pointer. Writing 1 to bit *n* resets receive channel *n* FIFO pointer. • 0b – No effect • 1b – Reset receive data channel *n* FIFO
23-18 —	Reserved
17-16 RCE	Receive Channel Enable Enables the corresponding data channel for receive operation. Changing this field takes effect immediately for generating the FIFO request and warning flags. It takes effect at the end of each frame for receive operations. The width of RCE = the number of receive channels (call it *n*). For example, if RCE is two bits wide, then bit position 16 refers to receive channel 1 and bit position 17 refers to receive channel 2. Writing 1 to bit 16 enables receive channel 1, and writing 1 to bit 17 enables receive channel 2. Writing 1 to bit *n* enables receive channel *n*. **NOTE** When there is only a single channel, you do not need an individual channel FIFO reset (RCR3[CFR]). In that case, use the global FIFO reset (RCSR[FR]). • 0b – Disable receive data channel *n* • 1b – Enable receive data channel *n*
15-5 —	Reserved
4-0 WDFL	Word Flag Configuration Configures which word sets the word start flag (RCSR[WSF]). The value must be one less than the word number (for example, write zero to select the first word in the frame). If you configure this field to a value greater than RCR4[FRSZ], RCSR[WSF] is never set. 0_0000b - Word 1 0_0001b - Word 2 0_0010b-1_1110b - Word (WDFL value + 1) 1_1111b - Word 32

67.6.1.17 Receive Configuration 4 (RCR4)

Offset

Register	Offset
RCR4	98h

Function

Contains the receive fields for FIFO Combine mode, FIFO Packing mode, and frame sync settings.

NOTE

Do not alter this register when RCSR[RE] is 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			FCONT	FCOMB		FPACK		0			FRSZ				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			SYWD					0		Reserved	MF	FSE	ONDEM	FSP	FSD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 FCONT	FIFO Continue on Error Configures when SAI continues receiving data after a FIFO error is detected. 0b - From the start of the next frame after the FIFO error flag is cleared 1b - From the same word that caused the FIFO error to become 1 after the FIFO warning flag is cleared
27-26 FCOMB	FIFO Combine Mode Enables FIFO Combine mode for specified operations. When FIFO Combine mode is enabled for FIFO reads, software reading any FIFO data register alternates the read among the enabled data channel FIFOs. For example, if two data channels are enabled:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• The first read is performed to the first enabled data channel FIFO. • The second read is performed to the second enabled data channel FIFO. Resetting the FIFO or disabling FIFO Combine mode for FIFO reads resets the pointer back to the first enabled data channel. When FIFO Combine mode is enabled for FIFO writes from the receive shift registers, the first enabled data channel input alternates between the enabled data channel FIFOs. For example, if two data channels are enabled: • The first unmasked received word is stored in the first enabled data channel FIFO. • The second unmasked received word is stored in the second enabled data channel FIFO. Because the first word of the frame is always stored in the first enabled data channel FIFO, NXP recommends that the number of unmasked words per frame is evenly divisible by the number of enabled data channels. 00b - Disable 01b - Enable on FIFO writes (from receive shift registers) 10b - Enable on FIFO reads (by software) 11b - Enable on FIFO writes (from receive shift registers) and reads (by software)
25-24 FPACK	FIFO Packing Mode Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8 bits or 16 bits, only the first 8 bits or 16 bits are stored in the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO read pointer only increments when software reads the full 32-bit FIFO word. 00b - Disable 01b - Reserved 10b - Enable 8-bit FIFO packing 11b - Enable 16-bit FIFO packing
23-21 —	Reserved
20-16 FRSZ	Frame Size Configures the number of words in each frame. The value must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words. 0_0000b - 1 0_0001b - 2 0_0010b-1_1110b - (FRSZ value + 1) 1_1111b - 32
15-13	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
12-8 SYWD	Sync Width Configures the length of the frame sync in number of bit-clock cycles. The value must be one less than the number of bit-clock cycles. For example, write 0 for the frame sync to assert for one bit-clock cycle only. You cannot configure this field to be longer than the first word of the frame. 0_0000b - 1 0_0001b - 2 0_0010b-1_1110b - (SYWD value + 1) 1_1111b - 32
7-6 —	Reserved
5 —	Reserved Write only zero to this field.
4 MF	MSB First Configures what is received first: LSB or MSB. 0b - LSB 1b - MSB
3 FSE	Frame Sync Early Determines when frame sync is asserted. 0b - First bit of the frame 1b - One bit before the first bit of the frame
2 ONDEM	On-Demand Mode Determines when the internal frame sync is generated. When this field is 1, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is 0. 0b - Generated continuously 1b - Generated when the FIFO warning flag is 0
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0b - Active high 1b - Active low
0 FSD	Frame Sync Direction Configures the direction of the frame sync.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Generated externally in Target mode 1b - Generated internally in Controller mode

67.6.1.18 Receive Configuration 5 (RCR5)

Offset

Register	Offset
RCR5	9Ch

Function

Contains receive word and bit index settings.

NOTE

Do not alter this register when RCSR[RE] is 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				WNW				0				W0W			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			FBT					0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28-24 WNW	Word N Width Configures the number of bits in each word except for the first one in the frame. The value must be one less than the number of bits per word. SAI does not support word widths less than 8 bits. SAI does not support bit settings 0–6 (0b–0_0110b).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0111b - 8 0_1000b - 9 0_1001b-1_1110b - (WNW value + 1) 1_1111b - 32
23-21 —	Reserved
20-16 W0W	Word 0 Width Configures the number of bits in the first word of each frame. The value must be one less than the number of bits in the first word. SAI does not support word widths less than 8 bits when there is only one word per frame. 0_0000b - 1 0_0001b - 2 0_0010b-1_1110b - (W0W value + 1) 1_1111b - 32
15-13 —	Reserved
12-8 FBT	First Bit Shifted Configures the bit index for the first bit received for each word in the frame. If configured for MSB-first, the index of the next bit received is one less than the current bit received. If configured for LSB-first, the index of the next bit received is one more than the current bit received. The value must be greater than or equal to the word width when configured for MSB-first. The value must be less than or equal to 31-word width when configured for LSB-first. See Data alignment for details. 0_0000b - 0 0_0001b-1_1110b - FBT value 1_1111b - 31
7-0 —	Reserved

67.6.1.19 Receive Data (RDR0 - RDR1)

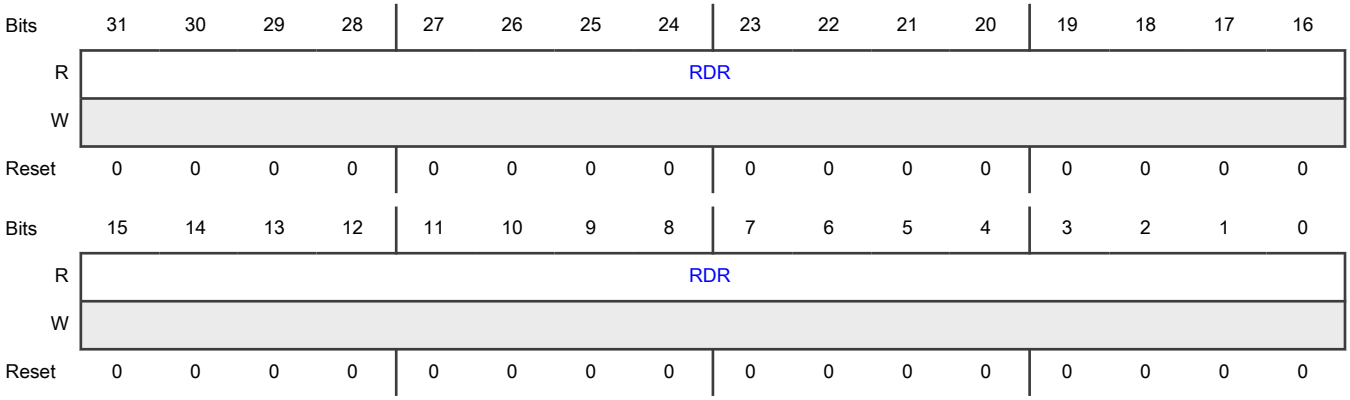
Offset

Register	Offset
RDR0	A0h
RDR1	A4h

Function
Contains receive data.

When the receive FIFO is not empty, reads from this register return the data from the top of the receive FIFO. When the receive FIFO is empty, SAI ignores reads from this register.

Diagram



Fields

Field	Function
31-0 RDR	Receive Data

67.6.1.20 Receive FIFO (RFR0 - RFR1)

Offset

Register	Offset
RFR0	C0h
RFR1	C4h

Function
Contains the receive FIFO pointers. The MSB of the read and write pointers distinguishes between FIFO full and empty conditions. If the read and write pointers are identical, the FIFO is empty. If the read and write pointers are identical except for the MSB, the FIFO is full.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												WFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCP	0											RFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-16 WFP	Write FIFO Pointer Indicates the FIFO write pointer for the receive data channel.
15 RCP	Read Channel Pointer Indicates whether this data channel is the next FIFO to be read when FIFO Combine mode is enabled for read operations. 0b - No effect 1b - Next FIFO to be read
14-4 —	Reserved
3-0 RFP	Read FIFO Pointer Indicates the FIFO read pointer for the receive data channel.

67.6.1.21 Receive Mask (RMR)

Offset

Register	Offset
RMR	E0h

Function

Contains the mask for the receive word. This register is double-buffered and updates:

- When RCSR[RE] first becomes 1.

- At the end of each frame.

This setup allows the masked words in each frame to change from frame to frame.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RWM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RWM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 RWM	Receive Word Mask Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame. 0000_0000_0000_0000_0000_0000_0000_0000b - Enable 0000_0000_0000_0000_0000_0000_0000_0001b - Mask

67.6.1.22 MCLK Control (MCR)

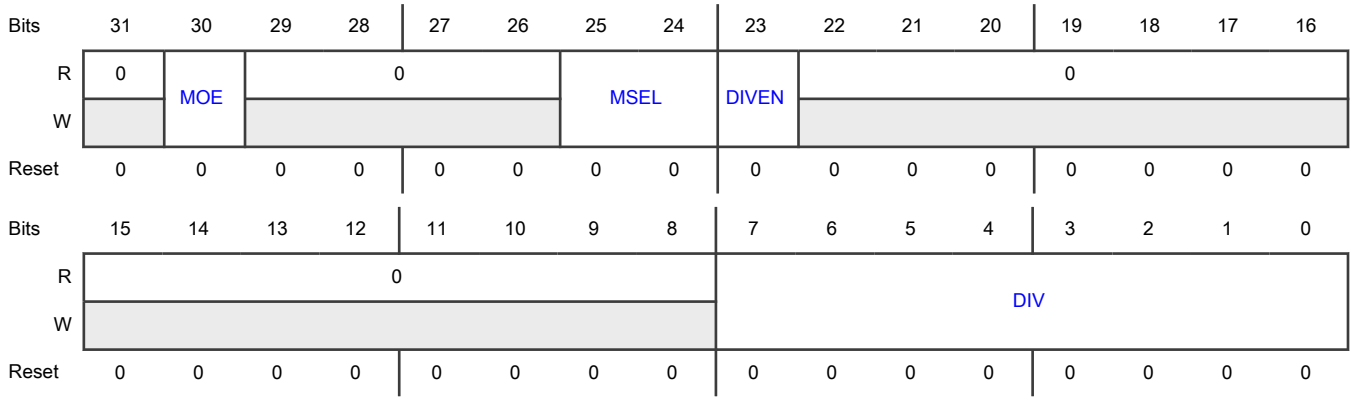
Offset

Register	Offset
MCR	100h

Function

Controls the audio controller clock.

Diagram



Fields

Field	Function
31 —	Reserved
30 MOE	MCLK Output Enable Configures the MCLK signal pin as an input or output. 0b - Input 1b - Output
29-26 —	Reserved
25-24 MSEL	MCLK Select Selects the clock used by the MCLK postdivider. You cannot change this field when the MCLK postdivider is enabled. See the chip-specific information for the connections to these inputs. 00b - Controller clock (MCLK) option 1 01b - Reserved 10b - Controller clock (MCLK) option 2 11b - Controller clock (MCLK) option 3
23 DIVEN	MCLK Post Divide Enable Enables the postdivider that is output on the MCLK signal pin. When enabled, the output on the MCLK signal pin is a postdivided version of the audio controller clock. When disabled, the output is the audio controller clock. The output divider does not impact the audio controller clock used by the transmitter or receiver. 0b - Disable 1b - Enable
22-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-0 DIV	MCLK Post Divide Specifies the postdivide value for the audio controller clock that only divides the output to the MCLK signal pin. The division value is (DIV + 1) × 2.

Chapter 68
Flexible Controller Area Network (FlexCAN)

68.1 Chip-specific CAN information

Table 582. Reference links to related information

Topic	Related module	Reference
Full description	CAN	CAN
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

68.1.1 Module instances

This device has two instances of FlexCAN module, CAN0 and CAN1.

68.1.2 Maximum data rate

The maximum data rate for FlexCAN in this device is 10 Mbps. Refer to device's Data sheet for more information.

68.2 Overview

FlexCAN is a communication controller implementing the CAN protocol according to the ISO 11898-1:2015 standard and CAN 2.0 Part B protocol specifications.

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific real-time processing and reliable operation requirements in the electromagnetic interference (EMI) environment of a vehicle. FlexCAN is a full implementation of the CAN protocol specification, the CAN with flexible data rate (CAN FD) protocol, and the CAN version 2.0 Part B protocol. It supports both standard and extended message frames and long payloads.

NOTE

Legacy Receive (RX) FIFO cannot be used in Flexible Data (FD) mode.

NOTE

In CAN FD mode, use the Enhanced Receive FIFO feature instead of the Legacy Receive FIFO.

68.2.1 Block diagram

Figure 393 shows the main submodules implemented in FlexCAN.

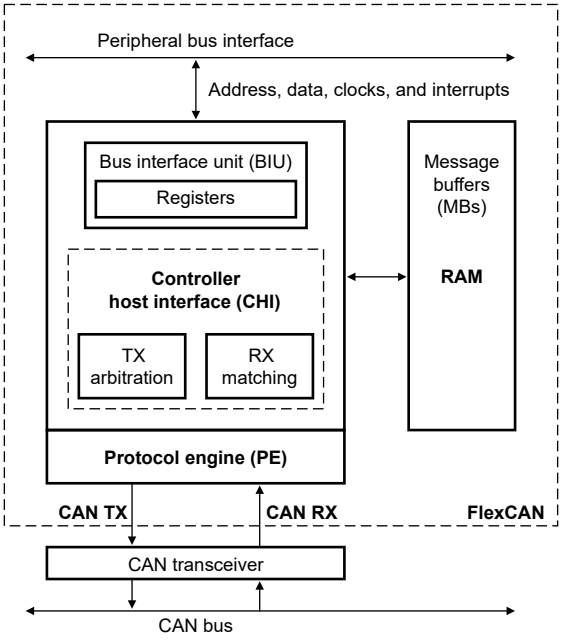


Figure 393. Block diagram

The Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- RAM access for receiving and transmitting message frames
- Receive message validation
- Error handling.
- CAN FD message detection

The Controller Host Interface (CHI) submodule manages:

- Message buffer (MB) selection for reception and transmission
- Arbitration and ID matching algorithms for both CAN FD and non-CAN FD message formats

The Bus Interface Unit (BIU) submodule controls access to and from the internal interface bus to establish connection to the CPU and to other blocks. The BIU manages access to:

- Clocks
- Address and data buses
- Interrupt outputs
- DMA
- Test signals

68.2.2 Features

- Full implementation of *CAN with Flexible data rate (CAN FD) protocol specification* and *CAN Specification Version 2.0, Part B*
  - Standard data frames
  - Extended data frames
  - Data length of 0–64 bytes
  - Programmable bit rate (see chip-specific FlexCAN information for specific maximum rate configuration)

  - Content-related addressing
- Compliance with ISO 11898-1:2015 standard
- Flexible message buffers that can be configured to store a payload of 0, 8, 16, 32, or 64 bytes
  - Increasing the payload size decreases the number of supported message buffers (see FlexCAN memory partition for CAN FD).
  - Message buffers are configurable as receive or transmit, supporting standard and extended messages.
- Individual Receive Mask registers for each message buffer
- Full-featured Legacy RX FIFO with storage capacity for six frames and automatic internal pointer handling with DMA support
- Full-featured Enhanced RX FIFO with storage capacity of 12 CAN FD frames and automatic internal pointer handling with DMA support
- Transmission abort capability
- Optional general purpose RAM space, using RAM not used by reception or transmission structures
- Listen-Only mode
- Programmable loopback mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Timestamp based on 16-bit free-running timer
- Global network time, synchronized by specific message
- Maskable interrupts
- Independence from transmission medium (external transceiver is assumed)
- Short latency time due to arbitration scheme for high-priority messages
- Low-power modes, with programmable wake-up on bus activity or matching with received frames (Pretended Networking)
- Transceiver delay compensation when transmitting CAN FD messages at faster data rates
- Management of remote request frames, automatically or by software
- Restriction only to write CAN bit time settings and configuration bits in Freeze mode
- Transmit message buffer status (lowest-priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- ESR1[SYNCH] to indicate module is synchronous with CAN bus
- CRC status for transmitted message
- Legacy RX FIFO Global Mask register
- Selectable priority between message buffers and receive FIFO during matching process
- Powerful Legacy RX FIFO ID filtering, capable of matching incoming IDs against either 128 extended IDs, 256 standard IDs, or 512 partial (8-bit) IDs, with 32 individual masking capability
- Powerful Enhanced RX FIFO ID filtering, capable of matching incoming IDs against either 16 extended or 32 standard ID filter elements with three filtering schemes: mask plus filter, range, and two filters without mask.
- Complete backward compatibility with previous FlexCAN version
- Pretended Networking functionality in low power: Stop mode

68.3 Functional description

FlexCAN is a CAN protocol engine with a flexible message buffer system for transmitting and receiving CAN frames. The system is a set of message buffers (MBs) that stores configuration and control data, timestamp, message ID, and data (see Message buffer structure).

For classical CAN frames, FlexCAN supports simultaneous reception through Legacy FIFO and message buffer. For CAN FD frames, FlexCAN supports reception through message buffer and enhanced receive FIFO.

For message buffer reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed in the ID field. A masking scheme makes it possible to match the ID programmed on the message buffer with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of message buffers to be transmitted based on the message ID (optionally augmented by three local priority bits) or the message buffer ordering.

A message buffer is active at a given time if it can participate in both the matching and arbitration processes. A receive message buffer with a 0b code is inactive (see Table 616). A transmit message buffer with a 1000b or 1001b code is also inactive (see Table 617).

FlexCAN can receive and transmit messages in CAN FD format. The message buffers are sized to store the quantity of data bytes selected by FDCTRL[MBDSRn]. The quantity of FD message buffers available for a given quantity of data bytes is described in the CAN FD Control (FDCTRL) register. See also FlexCAN memory partition for CAN FD.

68.3.1 Modes of operation

FlexCAN has these functional modes:

Table 583. Functional modes

Mode	Description
Normal	In Normal mode, FlexCAN receives and transmits message frames, manages errors normally, and enables all CAN Protocol functions.
Freeze	Freeze mode is enabled when MCR[FRZ] = 1. If enabled, FlexCAN enters Freeze mode when MCR[HALT] is 1 or when Debug mode is requested at chip level and FlexCAN writes 1 to MCR[FRZACK]. In this mode, no transmission or reception of frames is done, and synchronicity to the CAN bus is lost. See Freeze mode.
Loopback	FlexCAN enters this mode when CTRL1[LPB] becomes 1. In this mode, FlexCAN performs an internal loopback that can be used for self-test. The bit stream output of the transmitter is internally fed back to the receiver input. The receiving CAN input pin is ignored and the transmitting CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. To ensure proper reception of its own message, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field. Both transmit and receive interrupts are generated.
Listen-Only	FlexCAN enters this mode when CTRL1[LOM] becomes 1. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station are received. If FlexCAN detects an unacknowledged message, it flags a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.
CAN FD Active	In this mode, FlexCAN can transmit and receive all messages formatted according to the CAN FD Standard (2.0) and 2.0B Protocol in an interleaved fashion. The CPU can put FlexCAN into CAN FD Active mode by configuring MCR[FDEN] in Freeze mode.

Some features available in classical CAN are unavailable in CAN FD mode.

Table 584. Differences between classical CAN and CAN FD

Feature	Classical CAN	CAN FD
Legacy RX FIFO DMA	Yes	No
Legacy RX FIFO	Yes	No
Enhanced RX FIFO DMA	Yes	Yes
Enhanced RX FIFO	Yes	Yes
Pretended network support	Yes	No

FlexCAN can operate in these low-power modes:

Table 585. Low-power modes

Mode	Description
Module Disable	FlexCAN enters this mode when the CPU writes 1 to MCR[MDIS] and FlexCAN writes 1 to MCR[LPMACK]. After FlexCAN is disabled, it issues a request to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Writing 0 to MCR[MDIS] exits this mode. See Module Disable mode.
Stop	FlexCAN enters this mode when Stop mode is requested at chip level and FlexCAN writes 1 to MCR[LPMACK]. When in Stop mode, FlexCAN puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode occurs when the Stop mode request is removed, or when activity is detected on the CAN bus and the Self-Wake-up mechanism is enabled. See Stop mode.
Pretended Network (PN)	FlexCAN can operate in this mode with Stop mode. Before entering one of these low-power modes, you must write 1 to MCR[PNET_EN]. When in a low-power mode, CHI subblock clocks are shut down and CAN_PE subblock remains clocked. The receive process remains active to filter incoming messages (see Receive process in Pretended Networking mode) as defined by the configuration registers (see Pretended Networking Control 1 (CTRL1_PN)). Upon detecting a wake-up event, a wake-up interrupt is issued to the system. When MCR[PNET_EN] becomes 1, the CPU must disable Self-Wake-up by writing 0 to MCR[SLFWAK] (see Module Configuration (MCR).

68.3.1.1 Modes of operation details

FlexCAN has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following subsections contain functional details about Freeze mode and low-power modes.

CAUTION

FlexCAN does not support "Permanent Dominant" failure on the CAN bus line. If a Low-Power request or Freeze mode request occurs during a "Permanent Dominant" condition, the corresponding acknowledgment field can never be 1.

68.3.1.1.1 Freeze mode

This mode is requested either by the CPU writing 1 to MCR[HALT] or when the chip is put into Debug mode. MCR[FRZ] must be 1 and the module must not be in a low-power mode.

To obtain acknowledgment, FlexCAN writes 1 to MCR[FRZACK]. The CPU must only consider FlexCAN to be in Freeze mode when both request and acknowledgment conditions are satisfied.

When Freeze mode is requested, FlexCAN:

1. Waits to be in either Intermission, Error Passive, Bus Off, or Idle state.

2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in does not prevent entering Freeze mode.
3. Ignores the receive input pin and drives the transmit pin as recessive.
4. Stops the prescaler, halting all CAN protocol activities.
5. Grants write access to the Error Counters register, which is read-only in other modes.
6. Writes 1 to MCR[NOTRDY] and MCR[FRZACK].

After requesting Freeze mode, you must wait for MCR[FRZACK] to become 1 before executing any other action, otherwise FlexCAN may operate unpredictably. In Freeze mode, all memory-mapped registers are accessible.

Freeze mode is exited in one of these conditions:

- CPU writes 0 to MCR[FRZ].
- The chip is removed from Debug mode or the MCR[HALT] becomes 0.

MCR[FRZACK] becomes 0 after the protocol engine recognizes the negation of the freeze request. After leaving Freeze mode, FlexCAN tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

68.3.1.1.2 Module Disable mode

This low-power mode is normally used to disable a complete FlexCAN block temporarily, with no power consumption. The CPU requests this mode by writing 1 to MCR[MDIS], and FlexCAN acknowledges the request by writing 1 to MCR[LPMACK]. The CPU must only consider FlexCAN to be in Disable mode when both the request and acknowledgment conditions are satisfied.

If FlexCAN is disabled during Freeze mode, the module requests to disable the clocks to the PE and CHI submodules, writes 1 to MCR[LPMACK] and writes 0 to MCR[FRZACK].

It is not recommended to use Module Disable mode under Pretended Networking mode. Write 0 to MCR[MDIS] and wait for MCR[LPMACK] to become 0 before writing 1 to MCR[PNET_EN].

If the module is disabled during transmission or reception, FlexCAN:

1. Waits to be in either Idle or Bus Off state, or waits for the third bit of Intermission and then checks that it is recessive.
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. FlexCAN does not take a pending move-in into account.
3. Ignores its receive input pin and drives its transmit pin as recessive.
4. Shuts down the clocks to the PE and CHI submodules.
5. Writes 1 to MCR[NOTRDY] and MCR[LPMACK].

In this mode, the Bus Interface Unit continues to operate, enabling the CPU to access memory-mapped registers, except for:

- The Receive Message Buffers Global Mask registers
- The Receive Buffer 14 Mask register
- The Receive Buffer 15 Mask register

When in Disable mode, these items may not be accessed:

- The message buffers
- The Receive Individual Mask registers
- The reserved words within RAM

To exit this mode, the CPU writes 0 to MCR[MDIS], causing FlexCAN to request to resume the clocks and write 0 to MCR[LPMACK]. This write occurs after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

68.3.1.1.3 Stop mode

In this low-power mode for the system, all chip clocks can be stopped for maximum power savings. Stop mode is globally requested by the CPU and FlexCAN writes 1 to a Stop Acknowledgment signal to indicate acknowledgment. The CPU must only consider FlexCAN to be in Stop mode when both request and acknowledgment conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it shuts down the clocks globally by:

1. Writing 1 to MCR[LPMACK].
2. Writing 0 to MCR[FRZACK].
3. Sending the Stop Acknowledge signal to the CPU.

If Stop mode is requested during transmission or reception, FlexCAN:

1. Waits to be in either Idle or Bus Off state, or waits for the third bit of Intermission and verifies that it is recessive.
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. FlexCAN does not take a pending move-in into account.
3. Ignores its receive input pin and drives its transmit pin as recessive.
4. Writes 1 to MCR[NOTRDY] and MCR[LPMACK].
5. Sends a Stop Acknowledge signal to the CPU, so that the CPU can shut down the clocks globally.

FlexCAN exits Stop mode when the CPU resumes the clocks and removes the Stop mode request. This exit can occur as a result of the Self-Wake mechanism.

In Self-Wake, if MCR[SLFWAK] = 1 when FlexCAN enters Stop mode, then upon detecting a recessive-to-dominant transition on the CAN bus, FlexCAN writes 1 to ESR1[WAKINT]. If enabled by MCR[WAKMSK], FlexCAN generates a wake-up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Stop mode request. FlexCAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, FlexCAN does not receive the frame that woke it up. Table 586 details the effect of MCR[SLFWAK] and MCR[WAKMSK] upon wakeup from Stop mode. Waking from Stop mode only works when both fields are 1.

After the CAN protocol engine recognizes the negation of the Stop mode request, FlexCAN writes 0 to MCR[LPMACK]. FlexCAN then waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, FlexCAN does not receive the frame that woke it up.

Table 586. Waking from Stop mode

SLFWAK	WAKINT	WAKMSK	Chip clocks enabled	Wake-up interrupt generated
0	—	—	No	No
0	—	—	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	No	No
1	1	1	Yes	Yes

NOTE

When FlexCAN is in the Bus Off state and Low-Power mode, FlexCAN may take an extra 128 IDLE cycles to leave the Bus Off state after exiting Low-Power mode.

68.3.1.1.4 Pretended Networking (PN) mode

This special low-power mode receives wake-up messages with low power consumption. This mode can be selected to operate together with Stop mode. Before entering a low-power mode, MCR[PNET_EN] must be 1. After entering a low-power mode, the CHI subblock is shut down and the CAN_PE subblock is kept active. The receive process is still active to filter incoming messages (see Receive process in Pretended Networking mode) as defined by the configuration registers (see Pretended Networking Control 1 (CTRL1_PN)). Upon detecting a wake-up event, FlexCAN issues a wake-up interrupt to the system.

To enter Pretended Networking mode, FlexCAN must be in Normal mode, not in Freeze or Module Disable mode. In Pretended Networking mode, FlexCAN keeps itself synchronized with the CAN bus in Stop mode. Then, when Stop mode is requested, FlexCAN:

1. Waits to be in the Idle state, or waits for the third bit of Intermission, then verifies that it is recessive.
2. Writes 1 to MCR[LPMACK].
3. Requests the shutdown of the CHI submodule clock, while keeping the PE submodule clock active.

FlexCAN can exit Pretended Networking mode in one of these ways:

- The CPU removes the Stop mode request.
- FlexCAN waits until Bus Idle, or until the third bit of Intermission state, to write 0 to MCR[LPMACK].

The above exit events can be triggered:

- By FlexCAN, after detecting a wake-up event and issuing the respective interrupt.
- By the CPU itself, when another method wakes it up.

Consequently, FlexCAN waits until the Bus Idle state, or until the third bit of the Intermission state, to write 0 to MCR[LPMACK] and resume Normal mode. This procedure ensures that FlexCAN is synchronized to the CAN bus after exiting Pretended Networking mode. The CPU must wait for MCR[LPM_ACK] to become 0 before performing any access to FlexCAN.

When MCR[PNET_EN] becomes 1, the CPU must disable the self-wake-up by writing 0 to MCR[SLFWAK] (see Module Configuration (MCR)).

NOTE

For more information about the difference between FD and non-FD regarding this feature, see Table 584.

68.3.2 Transmission process

NOTE

Instances of MB_CS in this topic refer to items in message buffers. See Message buffer structure for details.

To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt flag is set, and clear it if necessary.
2. If the message buffer is active (transmission pending), request an abort of the transmission. Write the ABORT code (1001b) to the CODE field of the Control and Status word.
3. Poll the IFLAG register until the corresponding IFLAG flag is set, or wait for the interrupt request (if enabled by the respective IMASK field).
4. Read the CODE field to identify whether the transmission was aborted or transmitted (see Transmission abort mechanism).
5. Clear the corresponding interrupt flag.
6. Write the ID register (containing the local priority if enabled via MCR[LPRIOEN]).
7. Write payload data bytes.
8. Configure the Control and Status word as needed.
    a. Set ID type via MB_CS[IDE].

b. Set Remote Transmission Request (if needed) via MB_CS[RTR].

c. If MCR[FDEN] = 1, configure MB_CS[EDL] and MB_CS[BRS]. For details about the relationship between the written value and transmitted value of MB_CS[ESI], see Table 597.[3]

d. Configure Data Length Code in bytes via MB_CS[DLC]. See Table 619 for detailed information.

e. To transmit the CAN frame, activate the message buffer by writing Ch to MB_CS[CODE].

NOTE

To maximize software performance, configure all the fields in the MB_CS word in a single 32-bit write operation. If the fields are configured in separate writes, MB_CS[CODE] must be the last write in the Control and Status word.

When the MB is activated, it participates in the arbitration process and its contents are eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission exceeds the MB payload size, FlexCAN completes the expected DLC by adding a constant CCh pattern.

After a successful transmission:

1. The value of the free-running timer is written into the timestamp field.
2. The CODE field in the Control and Status word is updated.
3. Both Cyclic Redundancy Check (CRCR) and CAN FD CRC (FDCRC) are updated.
4. A status flag is set in the Interrupt Flag register.
5. If allowed by the corresponding Interrupt Mask Register field, an interrupt is generated.

After transmission, the new CODE field depends on the code used to activate the message buffer (see Table 616 and Table 617 in Message buffer structure).

When the Abort feature is enabled (MCR[AEN] is 1), after the Interrupt flag is set for an MB configured as transmit buffer, the message buffer is blocked. The CPU cannot update the message buffer until the Interrupt Flag is cleared by the CPU. The CPU must clear the corresponding IFLAG flag before preparing this MB for a new transmission or reception.

NOTE

For backward compatibility (MCR[AEN] is 0), write the INACTIVE code (1000b) to the CODE field to deactivate the MB. In this case, the pending frame may be transmitted without notification (see Message buffer inactivation).

68.3.3 Arbitration process

The arbitration process scans the message buffers, searching for the transmission MB that holds the message to be sent at the next opportunity. This MB is called the arbitration winner. The scan starts from the lowest number MB and continues to the higher ones.

The arbitration process is triggered when at least one of the following occur:

- CRC field of the CAN frame arrives. The starting point depends on the value of CTRL2[TASD].
- Error Delimiter field of a CAN frame is in progress.
- Overload Delimiter field of a CAN frame is in progress.
- The winner of the arbitration is deactivated, and the CAN bus has not reached the first bit of the Intermission field.
- CPU writes to the Control and Status word of a winner MB, and the CAN bus has not reached the first bit of the Intermission field.
- CHI is in the Idle state and the CPU writes to the Control and Status word of any message buffer.
- FlexCAN exits Bus Off state.

[3] There is no need to write the ESI field; it is automatically transmitted as dominant by error active nodes and as recessive by error passive nodes. If FlexCAN is operating as a network gateway, however, the CPU writes MB_CS[ESI] according to the error status of the node that sent the message.

- FlexCAN leaves Freeze mode or a low-power mode.

If the arbitration process does not evaluate all message buffers before the CAN bus reaches the first bit of the Intermission field, the temporary arbitration winner is invalidated. FlexCAN will not compete for the CAN bus at the next opportunity.

The arbitration process selects the winner among the active transmission message buffers at the end of the scan according to the values of CTRL1[LBUF] and MCR[LPRIOEN].

See Arbitration process (continued) for more information about this process.

68.3.3.1 Lowest-number message buffer first

If CTRL1[LBUF] is 1, the first (lowest number) active transmission message buffer found is the arbitration winner. MCR[LPRIOEN] has no effect when CTRL1[LBUF] is 1.

68.3.3.2 Highest-priority message buffer first

If CTRL1[LBUF] is 0, the arbitration process searches for the active transmission message buffer with the highest priority. The frame of this message buffer has a higher probability of winning the arbitration on the CAN bus when multiple external nodes compete for the bus simultaneously.

The sequence of bits considered for this arbitration is called the arbitration value of the message buffer. The transmission message buffer with the lowest arbitration value among all transmission message buffers has the highest priority.

If two or more message buffers have equivalent arbitration values, the message buffer with the lowest number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIOEN].

68.3.3.2.1 Local priority disabled

If MCR[LPRIOEN] = 0, the arbitration value is built using the exact sequence of bits that would be transmitted in a CAN frame where local priority is disabled.

Table 587. Composition of the arbitration value when local priority is disabled

Format	Message buffer arbitration value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	— (18 bits)	— (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

68.3.3.2.2 Local priority enabled

To enable local priority, MCR[LPRIOEN] must be 1. In this case, the message buffer PRIO field (see Message buffer structure) is included at the very left of the arbitration value.

Table 588. Composition of the arbitration value when local priority is enabled

Format	Message buffer arbitration value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	— (18 bits)	— (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

Because the PRIO field is the most significant part of the arbitration value, message buffers with low PRIO values have higher priority than message buffers with high PRIO values. This priority is maintained regardless of the rest of their arbitration values.

The PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

68.3.3.3 Arbitration process (continued)

After the arbitration winner is found (see Arbitration process), its content is copied to a hidden auxiliary message buffer called a transmit serial message buffer (TX SMB). The TX SMB has the same structure as a normal message buffer, but is not user-accessible. This copy operation is called move-out. After it is done, write access to the Control and Status word of the corresponding MB is blocked (if MCR[AEN] = 1). Write access is restored in one of the following events:

- The CPU clears the corresponding IFLAG flag after the message buffer is transmitted.
- FlexCAN enters Freeze mode or Bus Off state.
- FlexCAN loses the bus arbitration, or there is an error during the transmission.

At the first opportunity window on the CAN bus, the message on the TX SMB is transmitted according to the CAN protocol rules.

The arbitration process can be triggered under the following conditions:

- During RX and TX frames from CAN CRC field to end of frame. The value of CTRL2[TASD] may be changed to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT = 124 to 128. The value of CTRL2[TASD] may be changed to optimize the arbitration start point.
- During Control and Status write by CPU in Bus Idle. The first Control and Status write starts the arbitration process, and a second Control and Status write during this same arbitration restarts the process. If other Control and Status writes are performed, the transmission arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and Bus Idle state starts, then an arbitration process is triggered. In this case, the first and second Control and Status writes in Bus Idle do not restart the arbitration process. If there is not enough time to finish arbitration in the Wait For Bus Idle state and the next state is Idle, the scan is not interrupted. That scan is completed during Bus Idle state. During this arbitration, a Control and Status write does not cause an arbitration restart.
- Deactivation of an arbitration winner during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the Wait For Bus Idle state). If a resynchronization occurs during Wait For Bus Idle, the arbitration process is restarted.

The arbitration process stops when:

- All message buffers are scanned.
- A transmission message buffer is found (if lowest buffer feature is enabled).
- An arbitration winner deactivates or aborts during any arbitration process.
- There is not enough time to finish the transmission arbitration process (for instance, when a deactivation is performed near the end of frame). In this case, the arbitration process is pending.
- An error or overload flag occurs in the bus.
- A low-power or Freeze mode request occurs in Idle state.

Arbitration is considered pending when:

- It is not possible to finish arbitration process in time.
- A Control and Status write occurs during arbitration, when that write is performed in an MB whose number is lower than the transmission arbitration pointer.
- Any Control and Status write occurs when there is no transmission arbitration process in progress.
- RX Match has updated an RX code to TX code.
- FlexCAN enters the Bus Off state.

If a Control and Status write is performed in the arbitration winner, a new process is restarted immediately.

If a Control and Status write is performed in an MB whose number is higher than the transmission arbitration pointer, the ongoing arbitration process scans this MB as normal.

68.3.4 Receive process

To be able to receive CAN frames into a message buffer, the CPU must prepare it for reception by executing the following steps:

1. If the message buffer is active (either TX or RX), deactivate it (see Message buffer inactivation), preferably with a safe deactivation (see Transmission abort mechanism).
2. Write the ID word into the message buffer.
3. To activate the message buffer, write the EMPTY code (0100b) to the CODE field of the Control and Status word. No setup is required for EDL, BRS, and ESI fields. The respective fields in the received message overwrite these fields.

After the MB is activated, it can receive frames that match the programmed filter. At the end of a successful reception, the move-in process updates the message buffer (see Move-in) as follows:

1. The received data field (up to eight bytes for Classical CAN message format and up to 64 bytes for CAN FD message format) is stored.
2. The received Identifier field is stored.
3. The value of the free-running timer when the second bit of the Identifier field of the frame is written into the Timestamp field of the MB.
4. The received SRR, IDE, RTR, EDL, BRS, ESI, and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 616 and Table 617 in Section Message buffer structure).
6. If allowed by the corresponding Interrupt Mask field, a status flag is set in the Interrupt Flag register and an interrupt is generated.

The recommended way for the CPU to service (read) the frame received in a message buffer is:

1. Read the Control and Status word of that message buffer.
2. Verify that the BUSY bit is 0, indicating that the message buffer is locked. If it is not 0, repeat step 1 until it becomes 0. See Message buffer lock mechanism.
3. Read the contents of the message buffer. After the message buffer is locked, FlexCAN move-in processes do not modify its contents. See Move-in.
4. Acknowledge the proper flag in the IFLAG registers.
5. Unlock the message buffer by reading the free-running timer.

To verify frame reception, the CPU should poll the status flag bit for that specific message buffer in one of the IFLAG registers, not the CODE field of that message buffer. Polling the CODE field does not work in this case. After a frame is received and the CPU services the message buffer (by reading the Control and Status word and unlocking the message buffer), the CODE field does not return to EMPTY. It remains FULL, as explained in Table 616. If the CPU tries to work around this behavior by writing to the Control and Status word to force an EMPTY code after reading the message buffer without a prior safe deactivation, a newly received frame matching the filter of that message buffer may be lost.

CAUTION

In summary: never poll by reading the Control and Status word of the message buffers directly. Instead, read the IFLAG registers.

The Identifier field of the received frame is always stored in the matching message buffer. If the match was due to masking, the contents of the ID field in a message buffer may change. When MCR[SRXDIS] becomes 1, FlexCAN does not store frames transmitted by itself in any MB, even if it contains a matching receive MB. Also, no interrupt flag or interrupt signal is generated. Otherwise, when MCR[SRXDIS] becomes 0, if a matching receive MB exists, FlexCAN can receive frames transmitted by itself.

When MCR[DMA]) becomes 1, upon receiving a frame in the Legacy FIFO, IFLAG1[BUF5I] generates a DMA request and does not generate a CPU interrupt (see Legacy RX FIFO in DMA Operation). The IMASK1 fields in the Legacy RX FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the Control and Status word (read 80h address, optional).
2. Read the ID field (read 84h address, optional).
3. Read all data bytes (start read at 88h address, optional).

68.3.5 Matching process

The matching process scans the message buffer memory for RX MBs programmed with the same ID as the one received from the CAN bus. . The matching starts from the lowest number message buffer and continues toward the higher-numbered ones.

For enhanced RX FIFO, see Enhanced RX FIFO.

For legacy RX FIFO, see Legacy RX FIFO.

As the frame is received, it is stored in a hidden auxiliary MB called Receive Serial Message Buffer (RX SMB).

The starting point of the matching process depends on the following conditions:

- If the received frame is a remote frame, the starting point is the CRC field of the frame.
- If the received frame is a data frame with DLC field equal to zero, the starting point is the CRC field of the frame.
- If the received frame is a data frame and the DLC field has a nonzero value, the starting point is the DATA field of the frame.

If a matching ID is found in one of the message buffers, the move-in process transfers the contents of the RX SMB to the matched MB. If any CAN protocol error is detected, no match results are transferred to the matched MB at the end of reception.

The matching process scans all matching elements of the the active receive MBs (CODE is EMPTY, FULL, OVERRUN, or RANSWER). The process searches for a successful comparison to the matching elements of the RX SMB that is receiving the frame on the CAN bus. The RX SMB has the same structure as a message buffer. The reception structures (message buffers) associated with the matching elements that had a successful comparison are the matched structures. The matching winner is selected at the end of the scan among those matched structures. The matching winner depends on conditions described in Table 589.

Table 589. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Message buffer	0	—	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY, FULL, or OVERRUN
Message buffer	0	—	1	cmp_msk	cmp_msk	cmp_msk	EMPTY, FULL, or OVERRUN
Message buffer	1	0	—	cmp	no_cmp	cmp	RANSWER
Message buffer	1	1	0	cmp	no_cmp	cmp_msk	EMPTY, FULL, or OVERRUN
Message buffer	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY, FULL, or OVERRUN

1. For message buffer structure, if SMB[IDE] is 1, the ID is 29 bits (ID Standard plus ID Extended). If SMB[IDE] is 0, the ID is 11 bits (ID Standard). For Legacy RX FIFO structure, the ID depends on IDAM.
2. cmp: Compares the RX SMB contents to the MB contents regardless of masks.
3. no_cmp: The RX SMB contents are not compared to the MB contents.
4. cmp_msk: Compares the RX SMB contents to MB contents, accounting for masks.

NOTE

For Enhanced RX FIFO, see Enhanced RX FIFO matching process.

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the message buffer is EMPTY.
- The CODE field of the message buffer is either FULL or OVERRUN, and it has already been serviced (the CPU has read the Control and Status word and unlocked it as described in Message buffer lock mechanism).
- The CODE field of the message buffer is either FULL or OVERRUN and an inactivation (see Message buffer inactivation) is performed.

The scan order for message buffers is from the matching element with the lowest number to the higher ones.

68.3.5.1 Matching priority

MCR[IRMQ] affects the matching winner search for MBs. If the field is 0, the matching winner is the first matched MB whether it is free-to-receive or not. If it is 1, the matching winner is selected according to this priority:

1. The first free-to-receive matched message buffer
2. The last non-free-to-receive matched message buffer

68.3.5.2 Special cases

If a non-safe MB inactivation (see Message buffer inactivation) occurs during the matching process and the inactivated MB is the temporary matching winner, the temporary matching winner is invalidated. The matching elements scan is not stopped and not restarted; it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist. In this case, a message may be lost.

Consider an example where:

- MCR[IRMQ] is 1.
- There are two message buffers with the same ID: the second and fifth MBs in the array.
- FlexCAN starts receiving messages with that ID.

When the first message arrives, the matching algorithm finds the first match in message buffer number 2. The code of this message buffer is EMPTY, so the message is stored in that MB. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive." It continues looking, finds MB number 5, and stores the message in that MB. If yet another message with the same ID arrives, the matching algorithm finds no matching free-to-receive MBs, so it overwrites the last matched message buffer (MB number 5). In doing so, it updates the CODE field of the message buffer to OVERRUN.

The ability to match the same ID in more than one MB can be used to implement a reception queue to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages are queued into the message buffers. The CPU can examine the Timestamp field of the message buffers to determine the order in which the messages arrived.

Matching a range of IDs is possible via ID acceptance masks. FlexCAN supports individual masking per message buffer (see Receive Individual Mask (RXIMR0 - RXIMR31)). During the matching algorithm, if a mask field is 1, the corresponding ID bit is compared. If the mask field is 0, the corresponding ID bit is a "don't care". Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed when the module is in Freeze mode; otherwise, the module blocks them.

FlexCAN also supports an alternate masking scheme with only RX Message Buffers Global Mask (RXMGMASK), Receive 14 Mask (RX14MASK), and Receive 15 Mask (RX15MASK) for backward compatibility with legacy applications. This alternate masking scheme is enabled when the MCR[IRMQ] = 0.

68.3.6 Receive process in Pretended Networking mode

Pretended Networking mode adds specific wake-up functionality in low-power mode (Stop mode). When Pretended Networking (PN) mode is enabled by writing 1 to MCR[PNET_EN], FlexCAN continues processing received CAN messages in low-power mode. FlexCAN is able to detect specific wake-up messages by filtering them against identifier (ID) and payload target values using preselected matching criteria.

NOTE

Wake-up functionality is not available for messages in CAN FD format. When in PN mode, CAN FD format messages are ignored.

PN registers are located in the 0B00h–0B7Ch address range and can be written only in Freeze mode. These registers are used for writing PN configuration (both control and target values) prior to entering PN mode. They are also used for reading wake-up flags and the received message ID and data when returning to Normal mode after wake-up. The CPU must wait for MCR[LPMACK] to become 0 before performing any access to FlexCAN PN registers.

PN control registers are described in Pretended Networking Control 1 (CTRL1_PN) and Pretended Networking Control 2 (CTRL2_PN). The control fields that configure the filtering criteria are:

- Payload filtering selection: CTRL*n*_PN[PLFS]
- ID filtering selection: CTRL*n*_PN[IDFS]
- Filtering combination selection: CTRL*n*_PN[FCS]

Table 590. PN target values

Value	Description
FLT_IDE	IDE target value used to filter the incoming message by its format (standard or extended)
FLT_RTR	RTR target value used to filter the incoming message by its type (data or remote frame)
FLT_DLC_HI and FLT_DLC_LO	Target DLC range used to filter the size of the payload of an incoming message
FLT_ID1	ID target value used to filter the incoming message ID (equal to, smaller than or equal to, greater than or equal to, or the lower limit value in an ID range)
FLT_ID2	ID target value used as the upper limit in an ID range
PL1	Payload target value used to filter the incoming message payload (equal to, smaller than or equal to, greater than or equal to, or the lower limit value in a payload range)
PL2	Payload target value used as the upper limit in a payload range

IDE, RTR, ID, and payload filters have their respective masks. The ones in these masks determine which bits are considered in equality comparisons. The zeroes in these masks determine which bits are don't care. ID and payload masks are used only for exact ID and exact payload comparisons.

68.3.6.1 ID filtering

The IDs of incoming messages can be filtered based on the following criteria:

- Match the exact ID value, found by comparing the ID field of the incoming message to the content of target Pretended Networking ID Filter 1 (FLT_ID1). The ID mask is used.
- Less than or equal to the maximum range of ID. That is, any message with ID value smaller than or equal to the content of target FLT_ID1 is accepted. The ID mask is not used.
- Greater than or equal to the minimum range of ID. That is, any message with ID value greater than or equal to the content of target FLT_ID1 is accepted. The ID mask is not used.

- Inside a range of IDs. Any message with an ID value greater than or equal to the content of target FLT_ID1 and smaller than or equal to the content of target Pretended Networking ID Filter 2 or ID Mask (FLT_ID2_IDMASK) is accepted. The ID mask is not used.

See CTRL1_PN[IDFS].

The above criteria for ID filtering must be coherent with FLT_IDE and FLT_RTR target values in Pretended Networking ID Filter 1 (FLT_ID1). Only RX frames that match the respective IDE and RTR bits to the contents of FLT_ID1[FLT_IDE] and FLT_ID1[FLT_RTR] are compared. When a range of IDs is selected (CTRL1_PN[IDFS] = 11), both FLT_ID1 and FLT_ID2 are referred to the same FLT_IDE and FLT_RTR fields in FLT_ID1.

The ID mask is applied only to the exact ID comparison filtering option (CTRL1_PN[IDFS] = 00) to determine which bits are considered in the comparison. For the exact match option, the mask can select any bit within the ID field. For maximum range, minimum range, and inside range comparisons, the ID mask is not considered.

The IDE and RTR masks are applied in both exact and range ID comparison filtering options to determine which bits are considered in comparison.

68.3.6.2 Payload filtering

Similar to the ID criteria, 64-bit data or payloads (PL) of incoming messages can be filtered based on the following criteria:

- A match with the exact payload value, found by comparing the payload field of the incoming message to the content of PL1. The payload mask is used.
- Less than or equal to the maximum range of payload. That is, any message with payload value smaller than or equal to the content of PL1 is accepted. The payload mask is not used.
- Greater than or equal to the minimum range of payload. That is, any message with payload value greater than or equal to the content of PL1 is accepted. The payload mask is not used.
- Inside a range of payloads. Any message with a payload value greater than or equal to the content of PL1 and smaller than or equal to the content of PL2 is accepted. The payload mask is not used.

See CTRL1_PN[PLFS].

The above criteria for payload filtering must be coherent with upper and lower limit values in Pretended Networking Data Length Code (DLC) Filter (FLT_DLC). The payload of an incoming message is filtered using the selected criteria only if the DLC value of the incoming message is inside a DLC range:

- Greater than or equal to FLT_DLC[FLT_DLC_LO] (lower limit)
- Lower than or equal to FLT_DLC[FLT_DLC_HI] (upper limit)

A DLC value outside the specified range results in a mismatch. If you configure FLT_DLC_LO = FLT_DLC_HI, only payloads of the specified quantity of bytes are filtered. DLC is not maskable.

When the inside range of payloads option is selected (CTRL1_PN[PLFS] = 11), both PL1 and PL2 are considered with the 8-byte data length. All data bytes excluded by the DLC of the received message are considered to have value zero.

The payload mask is only used in the exact match option (CTRL1_PN[PLFS] = 00). This mask determines which bits or bytes in the 8-byte data field of both incoming message and the contents of PL1 register are used for matching. Mask length must meet the expected range of DLC values. For maximum range, minimum range, and inside range comparisons, the payload mask is not considered.

When FlexCAN receives a remote frame and CTRL1_PN[FCS] is configured to select payload comparison, payload filtering is not considered and the comparison results in a mismatch.

68.3.6.3 Other filtering

Incoming messages can also be filtered based on the quantity and rate of message reception, specifically:

- Several messages that match the filtering criteria for ID or payload for a predefined number of times. This number can be from 1 to 255. See Pretended Networking Control 1 (CTRL1_PN).

- No message matching the filtering criteria for ID or payload up to a timeout trigger. That is, non-reception of a matching message for a defined quantity of time. See Pretended Networking Control 2 (CTRL2_PN).

When the counter reaches the predefined timeout value, FlexCAN can generate a wake-up timeout event from an internal timer with associated comparator circuitry capable of generating a timeout flag. This behavior is specified in CTRL2_PN[MATCHTO].

The above filtering criteria can be used together as follows:

- Message ID filtering only
- Message ID filtering and Payload filtering
- Message ID filtering only occurring *n* times
- Message ID filtering and Payload filtering occurring *n* times

The timeout counter runs concurrently with the reception filtering process. Both engines (timeout counter and message filtering) are independent. If an incoming message matches the selected filter criteria, the timeout counter continues counting until the CPU wakes up. If the timeout counter reaches the target value, the message filtering process continues to filter incoming messages until the CPU wakes up. WU_MTC[MCOUNTER] reports the number of matched messages that occurred in Pretended Networking mode up to the moment the CPU wakes up.

In Pretended Networking mode, the wake-up event that may occur sets the respective wake-up flag (see Pretended Networking Wake-Up Match (WU_MTC)):

- WU_MTC[WUMF] indicates a successful match meeting the selected filtering criteria.
- WU_MTC[WTOF] indicates a timeout trigger.

Any of these flags generates interrupts to the CPU, provided the respective mask bits are enabled (CTRL1_PN[WUMF_MSK] = 1 or CTRL1_PN[WTOF_MSK] = 1).

There are four Wake-up Message Buffers (WMBs) used to store incoming messages in Pretended Networking mode. Up to four messages can be stored (see Wake-Up Message Buffer (WMB0_CS - WMB3_CS)).

When CTRL1_PN[NMATCH] = 1, only one message is received if the filtering criteria are matched. This message is stored in WMB0.

If CTRL1_PN[NMATCH] is between two and four, WMB1, WMB2, and WMB3 store the second, third, and fourth matching messages, respectively.

If CTRL1_PN[NMATCH] is greater than four, the last four matching messages are stored in the WMBs. The WMB index indicates the arrival order. The last message is stored in WMB3.

Only the valid data bytes of the incoming match message are stored in the data field of WMBs. The non-valid data bytes are read as zero. If DLC = 0 and RTR = 1, the data field is filled with zeroes. In any of the above cases, the wake-up interrupt is generated only when the filtering criteria are completed and CTRL1_PN[WUMF_MSK] = 1.

When a non-match wake-up event occurs (timeout or external) and WU_MTC[MCOUNTER] ≥ 4, the message stored in WMB0 does not have valid content. WMB0 is used as a buffer for the current message in the CAN bus. Messages received during Pretended Networking mode do not have time stamps, and the respective field in the WMB structure must be ignored.

In low-power mode (Stop), all processes are shut down except for the PN functionality inside the CAN_PE subblock, which remains clocked by the oscillator clock (see Clock domains and restrictions). FlexCAN continues to receive incoming messages, but only compares them to the predefined target values according to the selected filtering criteria. The matching, arbitration, move-in, and move-out processes, normally available in Normal mode, are not performed in Pretended Networking mode.

FlexCAN in Pretended Networking reacts to messages on the CAN bus in the same manner as in Normal mode. It generates acknowledge bits, detect and count errors, and so on.

68.3.7 Move process

There are two types of move process: move-in and move-out.

68.3.7.1 Move-in

The move-in process is the copying of a message received by an RX SMB to an RX message buffer that has matched it. Each RX SMB has its own move-in process, but only one is performed at a given time. The move-in starts only when the message held by the RX SMB has a corresponding match (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or already gone past:
  - The second bit of Intermission field next to the frame that carried the message that is in the RX SMB.
  - The first bit of an overload frame next to the frame that carried the message in the RX SMB.
- There is no ongoing matching process.
- The CPU is not locking the destination message buffer.
- No move-in process from another RX SMB is ongoing. If more than one move-in process is to be started at the same time, both are performed and the newest process substitutes for the oldest.

The term "pending move-in" is used throughout the documentation and stands for a move-to-be that does not satisfy all of the above conditions.

If any of the following conditions is satisfied, the move-in is canceled and the RX SMB is able to receive another message:

- The destination message buffer is inactivated after the CAN bus has reached the first bit of the Intermission field next to the frame that carried the message. Also, its matching process has finished.
- There is a previous pending move-in to the same destination message buffer.
- The RX SMB is receiving a frame transmitted by FlexCAN itself and self-reception is disabled (MCR[SRXDIS] = 1).
- Any CAN protocol error is detected.

If the module enters Freeze or Low-Power mode, the pending move-in is not canceled. It remains on hold, waiting for Freeze and Low-Power mode to be exited and for the module to be unlocked. If a message buffer is unlocked during Freeze mode, the move-in occurs immediately.

The move-in process is FlexCAN executing the following steps:

1. Read all data words from the RX SMB in accordance with the selected payload size for the RX storage element.
2. Write all data words to the RX message buffer according to the selected payload size for the RX storage element. If the data size of the storage element is smaller than the original payload size described in the DLC field of the message, the payload is truncated. The high-order bytes that do not fit the destination size are lost.
3. Read the Control and Status and ID words from the RX SMB.
4. Write the Control and Status and ID words to the RX message buffer, and update the CODE field.

The move-in process is not atomic; the inactivation of the destination message buffer immediately cancels it (see Message buffer inactivation). In this case, the message buffer may remain partially updated, and therefore incoherent.

To alert the CPU that the message buffer content is temporarily incoherent, the BUSY Bit (least significant bit of the CODE field) of the destination message buffer becomes 1 during move-in.

68.3.7.2 Move-out

The move-out process is the copying of content from a TX message buffer to the TX SMB when a message for transmission is available (see Arbitration process). The move-out occurs in the following conditions:

- In the first bit of Intermission field
- During the Bus Off state, when TX Error Counter is in the 124 to 128 range
- During the Bus Idle state
- During the Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently outside the Bus Idle state. In Bus Idle, the move-out has the lowest priority of the concurrent memory accesses.

68.3.8 Data coherence

In order to maintain data coherency and proper FlexCAN operation, the CPU must obey the rules described in Transmission process and Receive process.

68.3.8.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU whether the transmission was aborted or the frame could not be aborted and was transmitted instead.

These primary conditions must be fulfilled in order to abort a transmission:

- MCR[AEN] must be 1.
- The first CPU action must be the writing of abort code (1001b) into the CODE field of the Control and Status word.

Active message buffers configured for transmission must be aborted before they can be updated. The write operation is blocked and the transmission is not disturbed when the abort code is written to:

- A message buffer currently being transmitted.
- A message buffer that was already loaded into the TX SMB for transmission.

In this case, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration.
- There is an error during the transmission.
- The module is put into Freeze mode.
- The module enters the Bus Off state.
- There is an overload frame.

If none of the conditions above are reached:

1. The message buffer is transmitted correctly.
2. The interrupt flag is set in the proper IFLAG register.
3. If enabled, an interrupt to the CPU is generated.

The abort request is automatically cleared when the interrupt flag is set. If only one of the above conditions is reached, the frame is not transmitted. In this case:

1. The abort code is written into the CODE field.
2. The interrupt flag is set in the proper IFLAG register.
3. Optionally, an interrupt is generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, the write operation is not blocked. The MB is updated and the interrupt flag is set. In this way, the CPU only needs to read the abort code to verify that the active MB was safely inactivated. In this case, although MCR[AEN] = 1 and the CPU wrote the abort code, the MB is inactivated and not aborted, because the transmission did not start yet. A message buffer is aborted only when the abort request is captured and kept pending until one of the previous conditions is satisfied.

68.3.8.2 Message buffer inactivation

Inactivation protects the message buffer against updates by FlexCAN internal processes. It allows the CPU to rely on message buffer data coherence after having updated it, even in Normal mode.

Inactivation of transmission message buffers must be performed only when MCR[AEN] = 0.

If a message buffer is inactivated, it does not participate in the arbitration process or the matching process until it is reactivated. See Transmission process and Receive process for detailed instructions on how to inactivate and reactivate a message buffer.

To inactivate a message buffer, the CPU must update its CODE field to INACTIVE (either 0b or 1000b).

Because you cannot synchronize the CODE field update with FlexCAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of an inactivated RX message buffer may be lost without notice. This loss can occur even if there are other message buffers with the same filter.
- A frame containing the message within an inactivated TX message buffer may be transmitted without setting the respective IFLAG flag.

To perform a safe inactivation and avoid the above consequences for TX message buffers, the CPU must use the transmission abort mechanism (see Transmission abort mechanism).

The inactivation automatically unlocks the message buffer (see Message buffer lock mechanism).

68.3.8.3 Message buffer lock mechanism

In addition to message buffer inactivation, FlexCAN uses a message buffer lock mechanism to maintain data coherence for the receive process. When the CPU reads the Control and Status word of an RX message buffer with codes FULL or OVERRUN, FlexCAN is configured to allow the CPU to read the whole message buffer in an atomic operation. FlexCAN sets an internal lock flag for that message buffer.

The lock is released in any of the following events:

- The CPU reads the free-running timer (global unlock operation).
- The CPU reads the Control and Status word of another message buffer, regardless of its code.
- The CPU writes into the Control and Status word. This procedure is not recommended for normal unlocking, because it cancels a pending move and may lose a received message.

The message buffer lock prevents a new frame from being written into the message buffer when the CPU is reading it.

NOTE

The locking mechanism applies only to RX message buffers that have a code other than INACTIVE (0b) or EMPTY[1] (0100b). TX message buffers cannot be locked.

Consider an example where:

- The second and the fifth message buffers of the array are programmed with the same ID.
- FlexCAN has already received and stored messages into these two message buffers.
- The CPU reads message buffer number 5 while another message with the same ID is arriving.

When the CPU reads the Control and Status word of message buffer number 5, this message buffer is locked. The new message arrives and the matching algorithm finds no free-to-receive message buffers, so it overrides message buffer number 5. This message buffer is locked, so the new message cannot be written to it. The message remains in the RX SMB until the message buffer is unlocked, and only then is it written to the message buffer.

If the message buffer remains locked and another new message with the same ID arrives, the new message overwrites the one in the RX SMB. There is no indication of lost messages in the CODE field of the message buffer or in Error and Status 1 (ESR1).

When the message is moved from the RX SMB to the message buffer, the BUSY bit on the CODE field becomes 1. If the CPU reads the Control and Status word and identifies that the BUSY bit is 1, it must wait until the BUSY bit becomes 0 to access the MB.

[1] In previous FlexCAN versions, reading the Control and Status word locked the message buffer even when CODE indicates it is EMPTY. This behavior is maintained when the IRMQ bit is 0.

NOTE

If the BUSY bit is 1 or the message buffer is empty, reading the Control and Status word does not lock the message buffer.

Inactivation takes precedence over locking. If the CPU inactivates a locked receive message buffer, then its lock status is negated and the message buffer is marked as invalid for the current matching round. Any pending message on the RX SMB is not transferred to the message buffer. A message buffer is unlocked when the CPU reads Free-Running Timer (TIMER) or the Control and Status word of another message buffer.

The lock and unlock mechanisms have the same functionality in Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. If unlocking occurs during a low-power mode, however, the move-in is postponed (see Modes of operation). Move-in takes place only when the module returns to Normal or Freeze mode.

68.3.9 Enhanced RX FIFO

FlexCAN supports an enhanced RX FIFO engine which can store up to 12 CAN FD messages. The region 2000h– 204Bh contains the output of the FIFO, which the CPU should read. To enable the enhanced RX FIFO, write 1 to ERFCR[ERFEN]. FlexCAN has two FIFO options, Legacy RX FIFO and Enhanced RX FIFO, but both options cannot be enabled at the same time. See Legacy RX FIFO for additional information.

To configure the enhanced RX FIFO watermark, write a value to ERFCR[ERFWM]. If ERFCR[ERFWM] is configured, the CPU is notified only if a minimum number of messages is stored in the FIFO. When the number of stored messages is greater than the value in ERFCR[ERFWM], the module sets ERFSR[ERFWMI]. Optionally, if MCR[DMA] or ERFIER[ERFWMIIE] are enabled, a DMA transfer or an interrupt can be triggered, respectively.

For the enhanced RX FIFO to receive, the CPU must execute the configuration procedure below. If the CPU must change any configurations of the Enhanced RX FIFO, the same procedure must be followed.

Prerequisites

MCR[RFEN] must be 0.

Procedure

Step	Purpose	Programming	Notes
1	Enter Freeze mode.	See Freeze mode.	—
2	If enhanced RX FIFO is not already enabled, enable it.	Write 1 to ERFCR[ERFEN].	—
3	Reset enhanced RX FIFO engine.	Write 1 to ERFSR[ERFCLR].	—
4	If the enhanced RX FIFO error flags are set, clear them.	Write 1 to these flags: • ERFSR[ERFUFW] • ERFSR[ERFOVF] • ERFSR[ERFWMI] • ERFSR[ERFDA]	—
5	Specify the total number of enhanced RX FIFO filter elements to be used in Enhanced RX FIFO reception.	Write the number to ERFCR[NFE].	—

Table continues on the next page...

Table continued from the previous page...

Step	Purpose	Programming	Notes
6	Specify the number of extended ID and standard ID filter elements to be used in Enhanced RX FIFO reception.	Write the number to ERFCR[NEXIF].	ERFCR[NEXIF] ≤ ERFCR[NFE] + 1.
7	If you are using DMA, enable DMA.	Write 1 to MCR[DMA].	—
8	If you are using DMA, specify the number of words to transfer for each Enhanced RX FIFO data element.	Write the number to ERFCR[DMALW].	—
9	Specify the Enhanced RX FIFO watermark.	Write the number to ERFCR[ERFWM].	If MCR[DMA] = 1, ERFCR[ERFWM] should be 0h.
10	If you are using interrupts, enable the interrupts.	Write 1 to the interrupt enables in Enhanced RX FIFO Interrupt Enable (ERFIER).	—
11	Configure the filter elements.	Write to the ERFFEL*n* registers.	ERFFEL*n* registers are implemented in RAM; you must explicitly initialize them before prior any reception.
12	Exit Freeze mode.	See Freeze mode.	—

There are two types of enhanced RX FIFO filter elements that can be stored in ERFFEL*n* registers: extended-ID filter elements and standard-ID filter elements. Each extended-ID filter element is stored in two ERFFEL*n* registers, and each standard-ID filter element is stored in one ERFFEL*n* register. ERFCR[NFE] defines the total number of Enhanced RX FIFO filter elements.

In addition, the filter memory space can be split into two regions: one for extended-ID filter elements and another for standard-ID filter elements, according to ERFCR[NEXIF]. Figure 394 shows how the enhanced RX filter elements are defined. See Enhanced RX FIFO matching process for information about the Enhanced RX FIFO matching process and filter element formats.

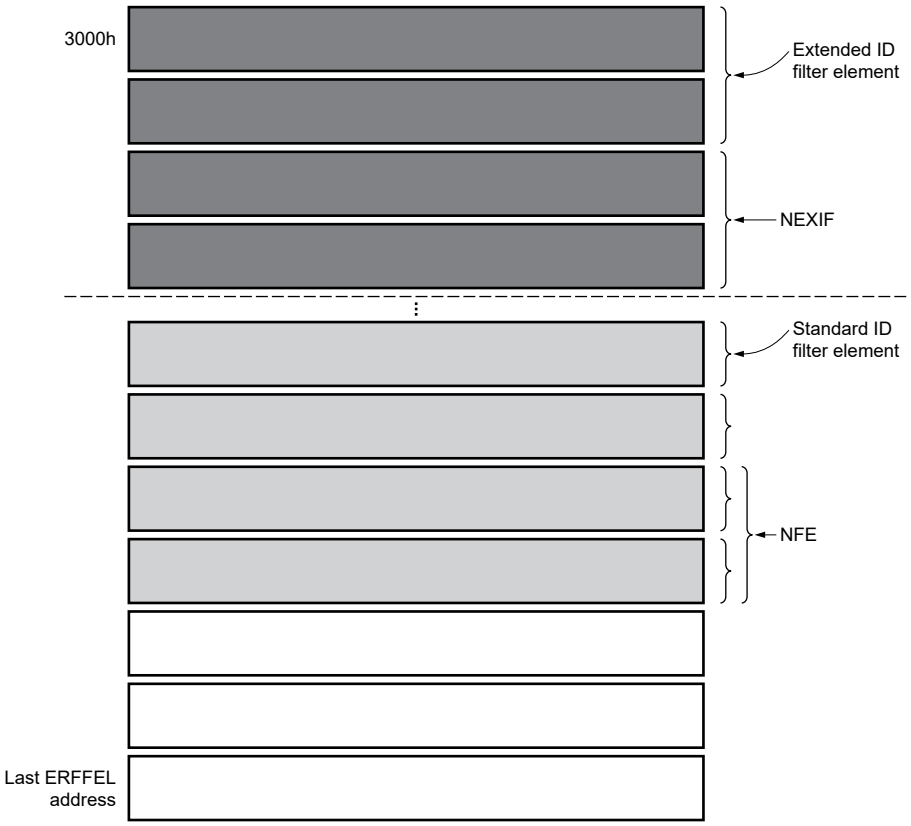


Figure 394. Number of enhanced RX FIFO filter elements

68.3.9.1 Enhanced RX FIFO matching process

When ERFCR[ERFEN] = 1, FlexCAN scans the ERFFEL*n* memory region. If at least one filter element satisfies the matching criteria, the CAN message content is transferred to the enhanced RX FIFO memory. If multiple filters match the incoming message ID, the first matching filter found by the matching process must be indicated in IDHIT.

Each ERFFEL*n* register can store one standard filter element. ERFFEL*n*[FEL][31:30], also called FSCH, determines the matching criteria in this way:

- If FSCH = b00, the filter scheme is based on mask and filter. A CAN message matches a standard ID filter element only if these criteria are reached:
  1. CAN message is base-frame format (IDE = 0).
  2. (ID[*n*] = STD ID filter [*n*]) or (STD ID Mask[*n*] = 0) for each bit *n* from 0 to 10.
  3. (RTR = RTR Filter) or (RTR MASK = 0).

In this explanation, RTR and ID are the Remote Transmit Request field and the ID from a CAN message, respectively.

If FSCH = b00, the filters and masks are defined as shown in Table 591.

Table 591. Standard ID filter element with filter and mask scheme (FSCH = b00)

31	30	29	28	27	26	16	15	12	11	10	0
FSCH = b00		Reserved		RTR filter	STD ID filter		Reserved		RTR mask	STD ID mask	

- If FSCH = b01, the filter scheme is based on range. A CAN message matches a standard ID filter element only if these criteria are reached:
  1. CAN message is base frame format (IDE = 0).
  2. ID ≥ STD ID Filter1.
  3. ID ≤ STD ID Filter2.
  4. (RTR = RTR filter) or (RTR MASK = 0).

RTR and ID are the Remote Transmit Request bit and ID from a CAN message, respectively. If FSCH = b01, the filters and mask are defined as shown in Table 592.

Table 592. Standard ID filter element with range scheme (FSCH = b01)

31	30	29	28	27	26	16	15	12	11	10	0
FSCH = b01		Reserved		RTRfilter	STD ID Filter2		Reserved		RTR mask	STD ID Filter1	

- If FSCH = b10, the filter scheme is based on two filters without masks. A CAN message matches a standard ID filter element only if these criteria are reached:
  1. CAN message is base frame format (IDE = 0).
  2. (ID[*n*] = STD ID Filter1[*n*]) or (ID[*n*] = STD ID Filter2[*n*]) for each bit *n* from 0 to 10.
  3. (RTR = RTR Filter1) or (RTR = RTR Filter2).

RTR and ID are the Remote Transmit Request bit and ID from a CAN message, respectively. If FSCH = b10, the filters are defined as shown in Table 593.

Table 593. Standard ID filter element with two-filter scheme (FSCH = b10)

31	30	29	28	27	26	16	15	12	11	10	0
FSCH = b10		Reserved		RTR Filter 2	STD ID Filter2		Reserved		RTR Filter 1	STD ID Filter1	

Each pair of ERFFEL*n* registers can store one extended filter element. ERFFEL*n*[FSCH] determines the matching criteria in this way:

- If FSCH = b00, the filter scheme is based on mask and filter. A CAN message matches an extended ID filter element only if these criteria are reached:
  1. CAN message is extended frame format (IDE = 1).
  2. (ID[*n*] = EXT ID filter [*n*]) or (EXT ID Mask[*n*] = 0) for each bit *n* from 0 to 28.
  3. (RTR = RTR Filter) or (RTR MASK = 0).

If FSCH = b00, the filters and masks are defined as shown in Table 594.

Table 594. Extended ID filter element with filter + mask scheme (FSCH = b00)

31	30	29	28	0
FSCH		RTR filter	EXT ID filter	
Reserved		RTR mask	EXT ID mask	

- If FSCH = b01, the filter scheme is based on range. A CAN message matches an extended ID filter element only if the following criteria are reached:
  1. CAN message is extended frame format (IDE = 1).
  2. ID ≥ EXT ID Filter1.
  3. ID ≤ EXT ID Filter2.
  4. (RTR = RTR Filter) or (RTR MASK = 0).

If FSCH = b01, the filters and masks are defined as shown in Table 595.

Table 595. Extended ID filter element with range scheme (FSCH = b01)

31	30	29	28																										0
FSCH		RTR filter	EXT ID Filter2																										
Reserved		RTR mask	EXT ID filter 1																										

- If FSCH = b10, the filter scheme is based on two filters without masks. A CAN message matches an extended ID filter element only if these criteria are reached:
  1. CAN message is extended frame format (IDE = 1).
  2. (ID[*n*] = EXT ID Filter1[*n*]) or (ID[*n*] = EXT ID Filter2[*n*]) for each bit *n* from 0 to 28.
  3. (RTR = RTR Filter1) or (RTR = RTR Filter2).

If FSCH = b10, the filters are defined as shown in Table 596.

Table 596. Extended ID filter element with two-filter scheme (FSCH = b10)

31	30	29	28																										0
FSCH		RTR Filter2	EXT ID Filter2																										
Reserved		RTR Filter1	EXT ID filter 1																										

68.3.9.2 Enhanced RX FIFO under DMA operation

You can enable the DMA feature by writing 1 to both ERFCR[ERFEN] and MCR[DMA]. The DMA controller can read the received message by reading a message buffer structure at the enhanced FIFO output port at the address range defined in Enhanced RX FIFO structure.

NOTE

FlexCAN supports 32-bit access only for DMA transfers.

For proper FIFO engine operation, the CPU should not access the Enhanced FIFO output port address range during DMA operation. Before writing 1 to MCR[DMA], the CPU must service Enhanced RX FIFO status bits. Otherwise, these bits may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before writing 0 to MCR[DMA], the CPU must first clear the ERFSR[ERFUFW], ERFSR[ERFOVF], ERFSR[ERFWMI], and ERFSR[ERFDA] flags. It must then clear the enhanced RX FIFO engine by writing one to ERFSR[ERFCLR].

When there is one frame available to be read from the Enhanced RX FIFO, FlexCAN sets ERFSR[ERFDA]. Upon receiving the request, the DMA controller can read the message in the Enhanced RX FIFO output. Each message reading process must end by the address defined in ERFCR[DMALW].

Follow these rules for Enhanced RX FIFO DMA operation:

- Because a DMA transfer cannot be changed dynamically, program ERFCR[DMALW] so the enhanced RX FIFO element can store the largest CAN message present on the CAN bus.
- Data bytes are valid according to the DLC field. See Table 619.

Each time the DMA controller reads one message from the FIFO, FlexCAN clears ERFSR[ERFDA]. If there is at least one message stored in the FIFO, FlexCAN sets it again.

Consider an example where the maximum number of bytes in the data field of a CAN frame for a certain application is eight. In that case, the last enhanced RX FIFO address offset can be found in Table 629 and Table 630. Using this address offset, ERFCR[DMALW] can be determined in this way:

- Maximum number of data bytes = 8
- Last address offset = TS_OFF = 2010h
- DMALW = 4

68.3.9.3 Enhanced RX FIFO clear operation

When ERFCR[ERFEN] is 1, the CPU can clear the Enhanced RX FIFO by writing 1 to ERFSR[ERFCLR]. The clear operation resets the internal FIFO pointers, but the FIFO content stored in RAM is not changed. This operation can only be performed in Freeze mode; the module blocks the operation in other modes. This operation does not clear ERFSR[ERFUFW], ERFSR[ERFOVF], ERFSR[ERFDA], or ERFSR[ERFWMI]. The CPU must service all these fields before executing the clear FIFO operation.

68.3.10 Legacy RX FIFO

The Legacy RX FIFO is receive-only. To enable it, write 1 to MCR[RFEN]. To maintain software backward compatibility with previous versions of FlexCAN that did not have the Legacy FIFO feature, the reset value of this field is zero.

CAUTION

Do not enable Legacy RX FIFO when the CAN FD feature is enabled.

The Legacy FIFO is six messages deep. The memory region the Legacy FIFO structure occupies (both message buffers and Legacy FIFO engine) is described in Legacy RX FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a message buffer structure at the output of the Legacy FIFO.

IFLAG1[BUF5I] (Frames Available in Legacy RX FIFO) is set when at least one frame is available to be read from the Legacy FIFO. If the corresponding mask bit enables it, an interrupt is generated. Upon receiving the interrupt, the CPU can read the message (accessing the output of the Legacy FIFO as a message buffer) and Legacy RX FIFO Information (RXFIR), then clear the interrupt. If there are more messages in the Legacy FIFO, clearing the interrupt:

1. Updates the output of the Legacy FIFO with the next message.
2. Updates RXFIR with the attributes of that message.
3. Reissues the interrupt to the CPU.

Otherwise, the flag remains cleared. The output of the Legacy FIFO is valid only when IFLAG1[BUF5I] is set.

IFLAG1[BUF6I] (Legacy RX FIFO Warning) is set when the Legacy RX FIFO receives a new message that increases the number of unread messages from four to five. This change means that the Legacy RX FIFO is almost full. The flag remains set until the CPU clears it.

IFLAG1[BUF7I] (Legacy RX FIFO Overflow) is set when an incoming message is lost because the Legacy RX FIFO is full. The flag is not set when the Legacy RX FIFO is full and a message buffer captures the message. The flag remains set until the CPU clears it.

Clearing one of the three flags above does not affect the state of the other two.

If an IFLAG flag is set and the corresponding mask bit is 1, an interrupt is generated.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing workload. The filtering criteria are specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN]. This table can be configured to one of the following formats (see also Legacy RX FIFO structure):

- Format A: 128 Identifier Acceptance Filters (IDAFs) — extended or standard IDs including IDE and RTR
- Format B: 256 IDAFs — standard IDs or extended 14-bit ID slices including IDE and RTR
- Format C: 512 IDAFs — standard or extended 8-bit ID slices

NOTE

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the Legacy RX FIFO has a corresponding Identifier Acceptance Filter Hit Indicator (IDHIT). The IDHIT can be read in the IDHIT field in the Control and Status word, as shown in the Legacy RX FIFO Structure description. The CPU can also obtain this information by accessing Legacy RX FIFO Information (RXFIR). RXFIR[IDHIT] refers to the message at the output of the Legacy FIFO, and is valid when IFLAG1[BUF5I] is set. RXFIR must be read only before clearing the flag, guaranteeing that the information refers to the correct frame within the Legacy FIFO.

The Individual Mask Registers (RXIMR*n*) individually affect up to 32 elements of the filter table, according to the value of CTRL2[RFFN]. This configuration allows very powerful filtering criteria to be defined. If MCR[IRMQ] is 0, the Legacy RX FIFO filter table is affected by Legacy RX FIFO Global Mask (RXFGMASK).

NOTE

See Table 584 for information about the difference between FD and non-FD regarding this feature.

68.3.10.1 Legacy RX FIFO in DMA Operation

The receive-only Legacy FIFO can support DMA. To enable this feature, write 1 to both MCR[RFEN] and MCR[DMA]. To maintain backward compatibility with previous versions of the module that did not have the DMA feature, the reset value of MCR[DMA] is zero.

The DMA controller can read the received message by reading a message buffer structure at the Legacy FIFO output port in the 80h–8Ch address range.

NOTE

FlexCAN supports 32-bit access only for DMA transfers.

When MCR[DMA] = 1, the CPU must not access the Legacy FIFO output port address range. Before writing 1 to MCR[DMA], the CPU must service the IFLAG flags set in the Legacy RX FIFO region. Otherwise, these flags may indicate that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before writing 0 to MCR[DMA], the CPU must perform a clear Legacy FIFO operation.

When at least one frame available to be read from the FIFO, IFLAG1[BUF5I] (Frames available in Legacy RX FIFO) is set. A DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the Legacy FIFO as a message buffer). The DMA reading process must end by reading address 8Ch. This read operation:

- Clears IFLAG1[BUF5I].
- Updates the FIFO output with the next message (if the FIFO is not empty).
- Updates Legacy RX FIFO Information (RXFIR) with the attributes of the new message.

If there are more messages stored in the FIFO, IFLAG1[BUF5I] is reasserted and another DMA request is issued. Otherwise, the flag remains cleared.

IFLAG1[BUF6I] and IFLAG1[BUF7I] are not used when the DMA feature is enabled.

When FlexCAN is working with DMA, the CPU does not receive any Legacy RX FIFO interruption and must not clear the related IFLAG flags. The related IMASK bits are not used to mask the generation of DMA requests.

NOTE

See Table 584 for information about the difference between FD and non-FD regarding this feature.

68.3.10.2 Clear Legacy FIFO

When MCR[RFEN] = 1, you can use the clear Legacy FIFO operation to empty Legacy FIFO contents. When the CPU writes 1 to IFLAG1[BUF0I], the clear FIFO operation occurs. This operation can only be performed in Freeze mode; FlexCAN blocks it in other modes. This operation does not clear the FIFO IFLAG flags; the CPU must service all FIFO IFLAG flags before executing the clear FIFO operation.

When Legacy RX FIFO is working with DMA, the clear FIFO operation clears IFLAG1[BUF5I], and the DMA request is canceled.

CAUTION

The clear Legacy FIFO operation does not clear IFLAG flags, except when MCR[DMA] = 1; in this case, only IFLAG1[BUF5I] is cleared.

68.3.11 CAN protocol-related features

68.3.11.1 CAN FD ISO compliance

The CAN FD protocol has been improved to increase the failure-detection capability that was in the original CAN FD protocol. This original protocol is also called non-ISO CAN FD, by CAN in Automation (CiA). A three-bit stuff counter and a parity bit have been introduced in the improved CAN FD protocol, now called ISO CAN FD. The CRC calculation has also been modified. All these improvements make the ISO CAN FD protocol incompatible with the non-ISO CAN FD protocol. FlexCAN still supports non-ISO CAN FD, so it can be used during an intermediate phase, for evaluation and development purposes.

It is recommended that you configure FlexCAN with the ISO CAN FD protocol by writing 1 to CTRL2[ISOCANFDEN].

68.3.11.2 CAN FD frames

ISO 11898-1:2015 specifies the Classical Frame format compliant to ISO 11898-1:2003 (2003) and introduces the CAN Flexible Data Rate Frame format (CAN FD). The Classical Frame format allows bit rates up to one Mbit/s and payloads up to eight bytes per frame. The Flexible Data Rate Frame format allows bit rates faster than one Mbit/s and payloads longer than eight bytes per frame. FlexCAN can receive and transmit CAN FD messages interleaved with Classical CAN messages.

There are additional control bits in the CAN FD frame:

- The Extended Data Length (EDL) bit enables a longer data payload with different data length coding.
- The Bit Rate Switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame.
- The Error State Indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes.

There are no Remote Frames (see Remote frames) in the CAN FD format. A message configured to transmit a Remote Frame is always sent out in Classical CAN format. When an FD frame is received and matches a message buffer, the RTR bit in the receiving message buffer becomes 0. The RTR bit must be considered in classical frames only.

68.3.11.2.1 CAN FD messages

CAN FD messages may be formatted as long frames where the data field exceeds eight bytes, and may range from 12 up to 64 bytes. They can also be configured to support bit rate switching. In this case, the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and end of the frame. Messages in Classical CAN format are limited to transport a maximum payload of eight bytes at nominal rate. Figure 395 illustrates the message formats for Classical and FD frames with either standard or extended ID.

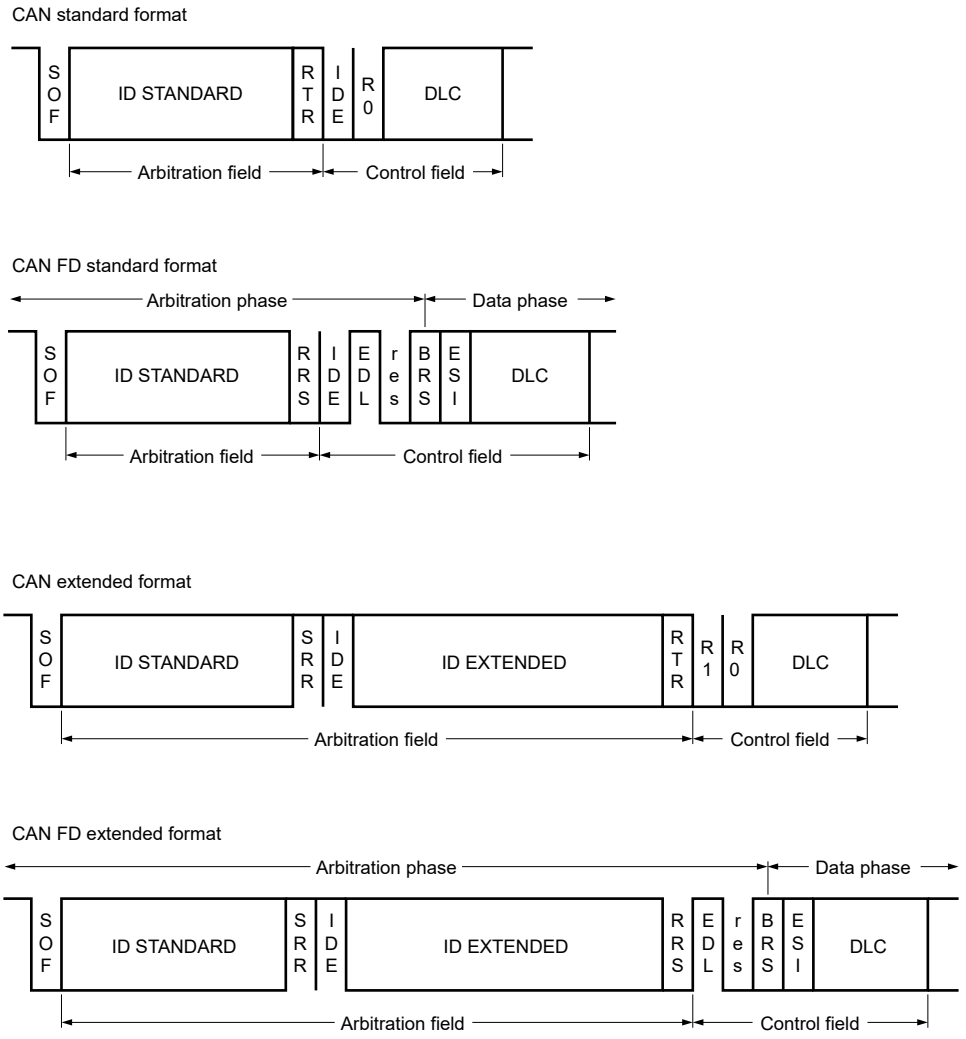


Figure 395. CAN message formats

MCR[FDEN] enables the ability to receive and transmit CAN FD messages. A recessive R0 bit in CAN frames with 11-bit identifiers, or a recessive R1 bit in CAN frames with 29-bit identifiers, is decoded as an EDL bit (not a reserved one). A recessive EDL bit identifies a CAN FD frame, and a dominant EDL bit identifies a Classical CAN frame. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded according to the DLC field value (see DLC definition in Message buffer structure).

CAN FD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the Start of Frame (SOF) bit until the Bit Rate Switch (BRS) bit is called the arbitration phase. This part is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC Delimiter bit, is called the data phase. When this second part is transmitted, a second set of CAN data bit timing configuration values determines the data bit rate. Finally, from the CRC delimiter until the Intermission bits, the transmission returns to nominal bit rate.

68.3.11.2.2 BRS in CAN FD

In CAN FD frames with bit rate switching, the bit timing changes inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by CAN Bit Timing (CBT). (Control 1 (CTRL1) also defines this timing for backward compatibility.) Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by CAN FD Bit Timing (FDCBT).

NOTE

If the time quantum length in nominal bit timing and in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase, and it lasts until the next synchronization event. The length of the time quantum should be the same in nominal and data bit timing. This configuration minimizes the chance of error frames on the CAN bus, and optimizes the clock tolerance in networks that use FD frames.

If BRS = 1 in the selected TX MB, FDCTRL[FDRATE] enables the transmission of all frames with bit rate switching. If FDCTRL[FDRATE] = 0, the transmission is performed at nominal rate regardless of the BRS bit value. FDCTRL[FDRATE] can be written at any time but takes effect only for the next message transmitted or received.

Nominal bit timing is resumed at the sample point of the CRC Delimiter bit or when an error is detected, whichever occurs first. Figure 396 describes the mechanism for entering and leaving the data phase when the BRS bit is recessive.

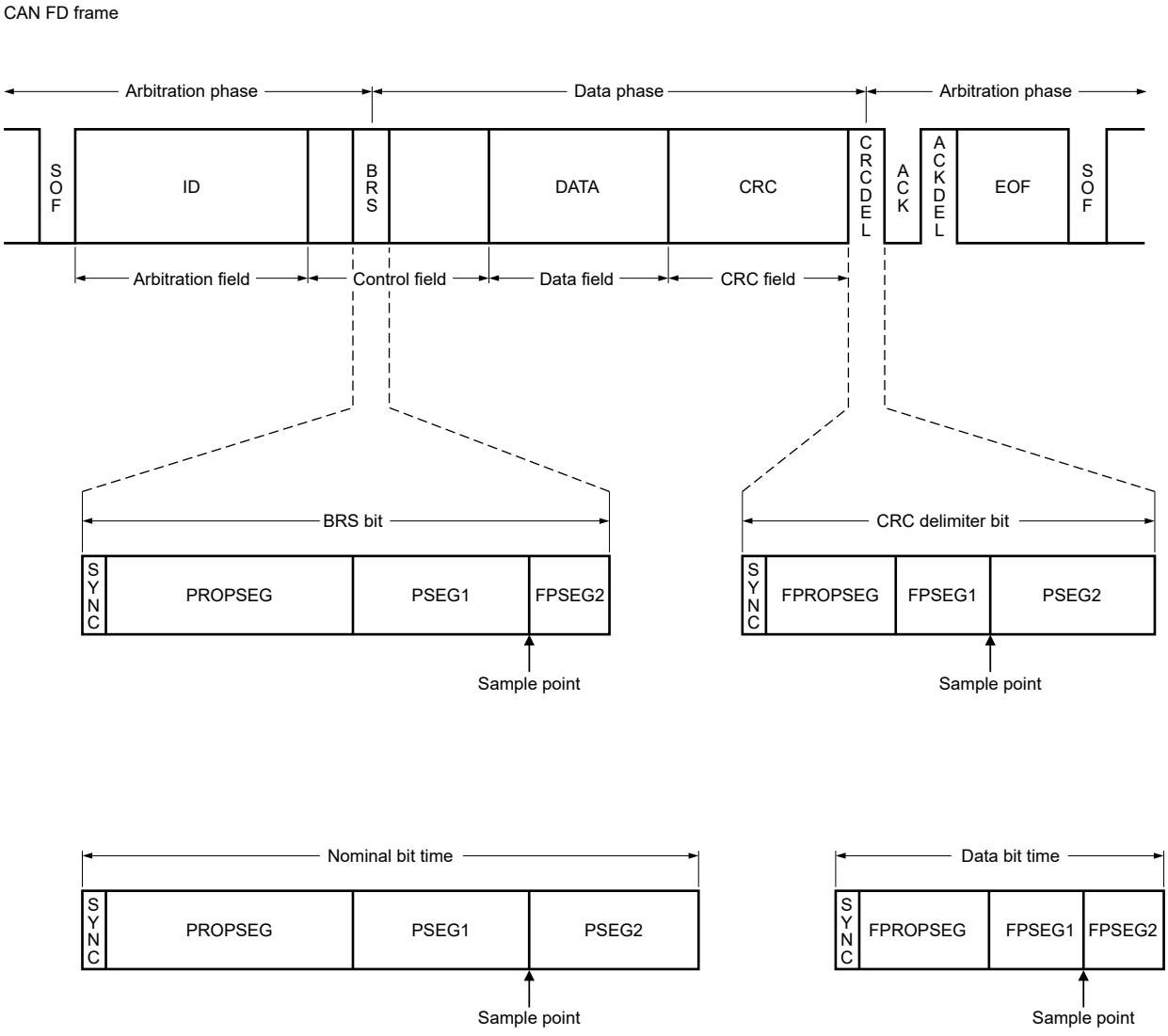


Figure 396. Bit rate switching mechanism for CAN FD messages

NOTE

In Classical CAN frames, the CRC delimiter is one recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. FlexCAN sends only one recessive bit as the CRC delimiter. It accepts two recessive bits before the recessive-to-dominant edge that starts the acknowledge slot. As a receiver, FlexCAN sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, FlexCAN accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The maximum configurable bit rate in the CAN FD data phase depends on the clock frequency of the CAN_PE subblock. For example, for a CAN_PE clock frequency of 40 MHz and the shortest configurable bit time of 5 time quanta, the bit rate in the data phase is 8 Mbit/s.

NOTE

The frequency used in this example may not be supported on this chip. It is shown only to demonstrate how the maximum configurable bit rate is calculated.

68.3.11.2.3 ESI in CAN FD

The value of the ESI bit is determined:

- By the error state of the transmitter at the start of the transmission, if the frame is originated in the FlexCAN node,
- Or by the original transmitting node when FlexCAN is acting as a gateway for the message.

If the transmitter is error-passive, ESI is transmitted recessive; otherwise, it is transmitted dominant. The permutations of the relationship between the written value and the transmitted value of the ESI are shown in Table 597.

Table 597. Written versus transmitted values of ESI field

FlexCAN fault confinement status at start of frame	ESI bit of TX MB	Transmitted ESI
Error active	0	0 (Error Active)
Error passive	0	1 (Error Passive)
Error active	1	1 (Error Passive)
Error passive	1	1 (Error Passive)

68.3.11.2.4 CRC calculations in CAN FD

Different CAN frame formats have different CRC polynomials. The first polynomial, CRC_15, is used for all frames in Classical CAN format. The second, CRC_17, is used for frames in CAN FD format with a data field up to 16 bytes long. The third, CRC_21, is used for frames in CAN FD format with a data field longer than 16 bytes. Each polynomial results in a Hamming distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The values of the EDL bit and the DLC field select the CRC sequence to be transmitted. When receiving a message, FlexCAN decodes EDL and DLC to select the adequate CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In Classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, CAN FD CRC (FDCRC) stores the calculated CRC for the transmitted message. This storage is performed with adequate length for the type of message, for CAN FD and non-FD messages. Cyclic Redundancy Check (CRCR) reports a valid CRC for Classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method changes for the CRC sequence, so the stuff bits are inserted at fixed positions. When FlexCAN is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence. This insertion occurs even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When FlexCAN receives a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A stuff error is detected if the fixed stuff bit has the same value as its preceding bit.

68.3.11.2.5 CAN FD errors

FlexCAN detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters ECR[RXERRCNT] and ECR[TXERRCNT] accumulate the counts of RX and TX errors, respectively, for both FD and non-FD frames indiscriminately. Two extra error counters, ECR[RXERRCNT_FAST] and ECR[TXERRCNT_FAST], accumulate RX and TX errors occurring in the data phase of CAN FD frames with BRS = 1 only. The rules for updating the error counters are the same for both CAN FD and non-FD frames (see Error Counter (ECR)).

These error flags report errors in both CAN FD and non-FD frames:

- ESR1[BIT1ERR]
- ESR1[BIT0ERR]
- ESR1[ACKERR]
- ESR1[CRCERR]
- ESR1[FRMERR]
- ESR1[STFERR]

If CTRL1[ERRMSK] = 1, they also generate the ERRINT interrupt.

These additional error flags indicate the occurrence of errors in the data phase of CAN FD frames with BRS = 1:

- ESR1[BIT1ERR_FAST]
- ESR1[BIT0ERR_FAST]
- ESR1[CRCERR_FAST]
- ESR1[FRMERR_FAST]
- ESR1[STFERR_FAST]

No ACKERR is detected in the data phase of a CAN FD frame. Fault confinement status reported in ESR1[FLTCONF] is the same for both CAN FD and Classical CAN frames, and is based on ECR[RXERRCNT] and ECR[TXERRCNT] only. Information in ECR[RXERRCNT_FAST] and ECR[TXERRCNT_FAST] may be considered as status to help detect the error nature related to the bit rate value.

When FlexCAN detects an error while transmitting or receiving a CAN FD message in the data phase, it immediately switches:

- Back to the arbitration phase, and
- Back to the nominal rate to start an error flag.

68.3.11.2.6 CAN FD synchronization

Resynchronization and hard synchronization occur in CAN FD frames in the same way as in Classical CAN ones. A hard synchronization is also performed at the recessive-to-dominant edge from EDL to R0 in CAN FD format frames. FlexCAN does not resynchronize when transmitting in the CAN FD data phase.

68.3.11.3 Transceiver delay compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase, when BRS = 1 in the message. This feature enables the use of rates up to 8 Mbit/s.

During the data phase of a CAN FD frame, if the transmitter cannot receive its own latest transmitted bit at the sample point of that bit, it detects a bit error. When bit rate switching is enabled (BRS = 1), the CAN bit time in the data phase can become shorter than the loop delay of the transceiver. This condition impedes the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

The transceiver delay compensation (TDC) process defines a secondary sample point where the transmitted bit is correctly compared to the received bit to check for bit errors.

You can enable the TDC mechanism via FDCTRL[TDCEN] or ETDC[ETDCEN]. The TDC mechanism is effective only during the data phase of FD frames with BRS = 1. It has no effect on either non-FD frames or FD frames transmitted at the normal bit rate. When the transmitted message has BRS = 1, TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit. When TDC is active, the real received bit is compared to the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

NOTE

The transmitters using TDC disregard the value of the CRC delimiter bit. A global error at the end of the CRC field causes the receivers to send error frames that the transmitter detects during Acknowledge or End of Frame.

For every transmitted FD frame with BRS = 1, the transition from the recessive EDL bit to the dominant R0 bit triggers the delay measurement (as shown in Figure 397). The loop delay is measured in Protocol Engine (PE) clock periods (CANCLK, see Protocol timing), from the transmitted EDL-R0 edge to the received EDL-R0 edge. The measured loop delay time added to an offset value specified in FDCTRL[TDCOFF] or ETDC[ETDCOFF] determines the position of the secondary sample point. FDCTRL[TDCVAL] or ETDC[ETDCVAL] stores the result of this calculation. The TDCVAL and ETDCVAL value saturates at its maximum value of 63 CANCLK and 255 CANCLK when the delay measurement is too long.

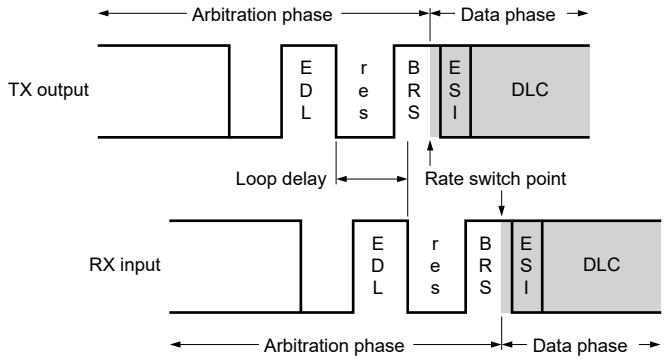


Figure 397. Transceiver loop delay measurement

The measured loop delay is not enough to define the secondary sample point, because it relates to the CAN bit edges. The transceiver delay compensation offset FDCTRL[TDCOFF] or ETDC[ETDCOFF] is used to shift the secondary sample point to an intermediate point inside the bit time, far away from its edges. The value of FDCTRL[TDCOFF] or ETDC[ETDCOFF] cannot be larger than the CAN bit duration in the data phase.

If the secondary sample point is set very near the CAN bit edge (SYNC field), problems may occur during the bit sampling in the data phase. For the TDC to work reliably, the offset must use optimal settings. To ensure that bit sampling is performed in the best region, configure the TDC offset as shown in this equation:

$$Offset = (FPSEG1 + FPROPSEG + 2) \times (FPRESDIV + 1)$$
$$or$$
$$Offset = (DTSEG1 + 2) \times (EDPRESDIV + 1), \text{ if } ETDCEN$$

Equation 29. TDC offset calculation

Figure 398 shows the SSP position when these settings are used.

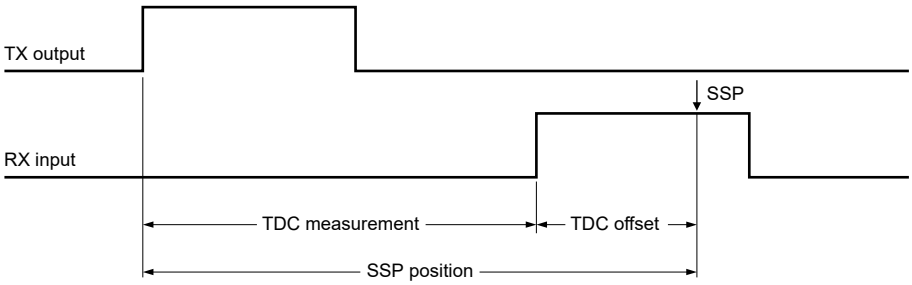


Figure 398. SSP position with optimal values

Alternatively, if CTRL2[BTE] and ETDC[ETDCEN] are 1, you can write 1 to ETDC[TDMDIS] to disable the transceiver delay measurement. In this case, only ETDC[ETDCOFF] defines the SSP position. Figure 399 shows the secondary sample point position when the transceiver delay measurement is disabled.

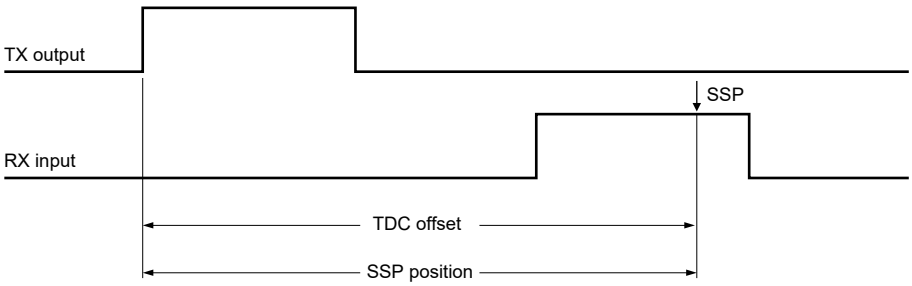


Figure 399. SSP position with TDC measurement disabled by ETDCEN and TDMDIS

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every TX CAN bit:

- The transmitted TX bit value is temporarily stored in a buffer.
- A time countdown based on FDCTRL[TDCVAL] or ETDC[ETDCVAL] is started. This countdown ends with the comparison of the received RX bit (delayed by the external loop delay plus the specified offset) to the stored TX bit.

If a bit error is detected at the secondary sample point, FlexCAN issues an error flag to the CAN bus at the next sample point.

During the arbitration phase, delay compensation is always disabled. During the data phase, the TDC mechanism of FlexCAN can compensate a maximum delay of 3 CAN bit times − 2 Tq. Beyond this limit, the FDCTRL[TDCFAIL] or ETDC[ETDCFAIL] flag is set. The flag indicates when the TDC mechanism is out of range and is unable to compensate the transceiver loop delay.

68.3.11.4 Remote frames

A remote frame is a special type of frame. You can program a message buffer to be a remote request frame by configuring the message buffer as Transmit with the RTR = 1. After the remote request frame is transmitted successfully, the message buffer becomes a receive message buffer, with the same ID as before.

When FlexCAN receives a remote request frame, the frame can be treated in different ways, depending on remote request storing (CTRL2[RRS]) and RX FIFO Enable (MCR[RFEN]):

- If RRS = 0, the ID of the frame is compared to the IDs of the transmit message buffers with the CODE field 1010b. If a matching ID exists, this message buffer frame is transmitted. If the matching message buffer has the RTR = 1, FlexCAN transmits a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response.

  The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. If a remote request frame is received and matches a message buffer, this message buffer immediately enters the internal arbitration process. However, it is considered a normal TX message buffer, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

- If CTRL2[RRS] = 1, the ID of the frame is compared to the IDs of the receive message buffers with the CODE field 0100b, 0010b, or 0110b. If a matching ID exists, this message buffer stores the remote frame in the same fashion as a data frame. No automatic remote response frame is generated. The mask registers are used in the matching process.
- If MCR[RFEN] = 1, FlexCAN does not generate an automatic response for remote request frames that match the Legacy FIFO filtering criteria. If the remote frame matches one of the target IDs, it is stored in the Legacy FIFO and presented to the CPU.

  For filtering formats A and B (see Legacy RX FIFO structure), it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted if they match the ID. Remote request frames are considered as normal frames. They generate a Legacy FIFO overflow when a successful reception occurs and the Legacy FIFO is already full.
- If ERFCR[ERFEN] = 1, FlexCAN does not generate an automatic response for remote request frames that match the Enhanced RX FIFO filtering criteria. Remote Request Frames are considered normal frames. They generate an Enhanced RX FIFO overflow when a successful reception occurs and the enhanced RX FIFO is already full.

NOTE

There is no remote frame in the CAN FD format. A fixed dominant RRS bit replaces the RTR bit. FlexCAN receives and transmits remote frames in the Classical CAN format.

68.3.11.5 Overload frames

When a dominant bit is detected on the CAN bus in these locations, FlexCAN transmits overload frames:

- The first or second bit of Intermission.
- The seventh bit (last) of End of Frame field (RX frames).
- The eighth bit (last) of Error Frame Delimiter or Overload Frame Delimiter.

68.3.11.6 Timestamp

The value of the free-running timer is sampled at the beginning of the Identifier field on the CAN bus. This value is stored at the end of move-in in the TIME_STAMP field, providing network behavior regarding time.

The FlexCAN bit clock clocks the free-running timer, which defines the baud rate on the CAN bus. During a message transmission or reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The free-running timer is not incremented during Disable, Stop, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See CTRL1[TSYN].

68.3.11.7 Protocol timing

Figure 400 shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule.

NOTE

To identify the proper clock source, see the clock distribution chapter (module clocks table).

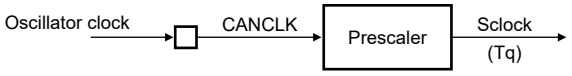


Figure 400. CAN engine clocking scheme

68.3.11.7.1 Bit timing configuration

FlexCAN supports various means to configure bit timing parameters required by the CAN protocol. Control 1 (CTRL1) has various fields to control bit timing parameters:

- CTRL1[PRESDIV]

- CTRL1[PROPSEG]
- CTRL1[PSEG1]
- CTRL1[PSEG2]
- CTRL1[RJW]

CAN Bit Timing (CBT) extends the range of the CAN bit timing variables in CTRL1. Enhanced Data Phase CAN Bit Timing (EDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) = 1.

Enhanced Nominal CAN Bit Timing (ENCBT) extends the range of CAN bit timing variables in CBT. Enhanced Nominal CAN Bit Timing (ENCBT) extends the range of CAN bit timing variables in FDCBT. When using ENCBT and EDCBT, you must program the nominal bit timing and data phase serial clock (Sclock) dividers in Enhanced CAN Bit Timing Prescalers (EPRS).

NOTE

When the CAN FD feature is enabled, always write 1 to CBT[BTF] or CTRL2[BTE] and specify the CAN bit timing variables in CBT or ENCBT. See CAN Bit Timing (CBT) or Enhanced Nominal CAN Bit Timing (ENCBT).

CTRL1[PRESDIV], and its extended range CBT[EPRESDIV] (or EPRS[ENPRESDIV]) and FDCBT[FPRESDIV] (or EPRS[EDPRESDIV]) for the data phase bits of CAN FD messages, defines the prescaler value that generates the serial clock (Sclock). (See Equation 30.) The period of Sclock defines the time quantum used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time managed by the CAN engine. It is the smallest time unit for all configuration values.

$$Tq = \frac{(PRESDIV + 1)}{f_{CANCLK}}$$

Equation 30. Time quantum

The bit rate, which defines the rate the CAN message is received or transmitted, is calculated with the formula:

$$CAN\ bit\ time = (Number\ of\ time\ quanta\ in\ 1\ bit\ time) \times Tq$$

$$Bit\ rate = \frac{1}{CAN\ bit\ time}$$

Equation 31. CAN bit time and baud rate

68.3.11.7.2 Bit time segments

A bit time is subdivided into three segments as shown in Figure 401. See also Figure 402 , Figure 403, and Table 598.

NOTE

For further explanation of the underlying concepts, see ISO 11898-1:2015. See also *CAN Specification Version 2.0, Part A and Part B* for bit timing.

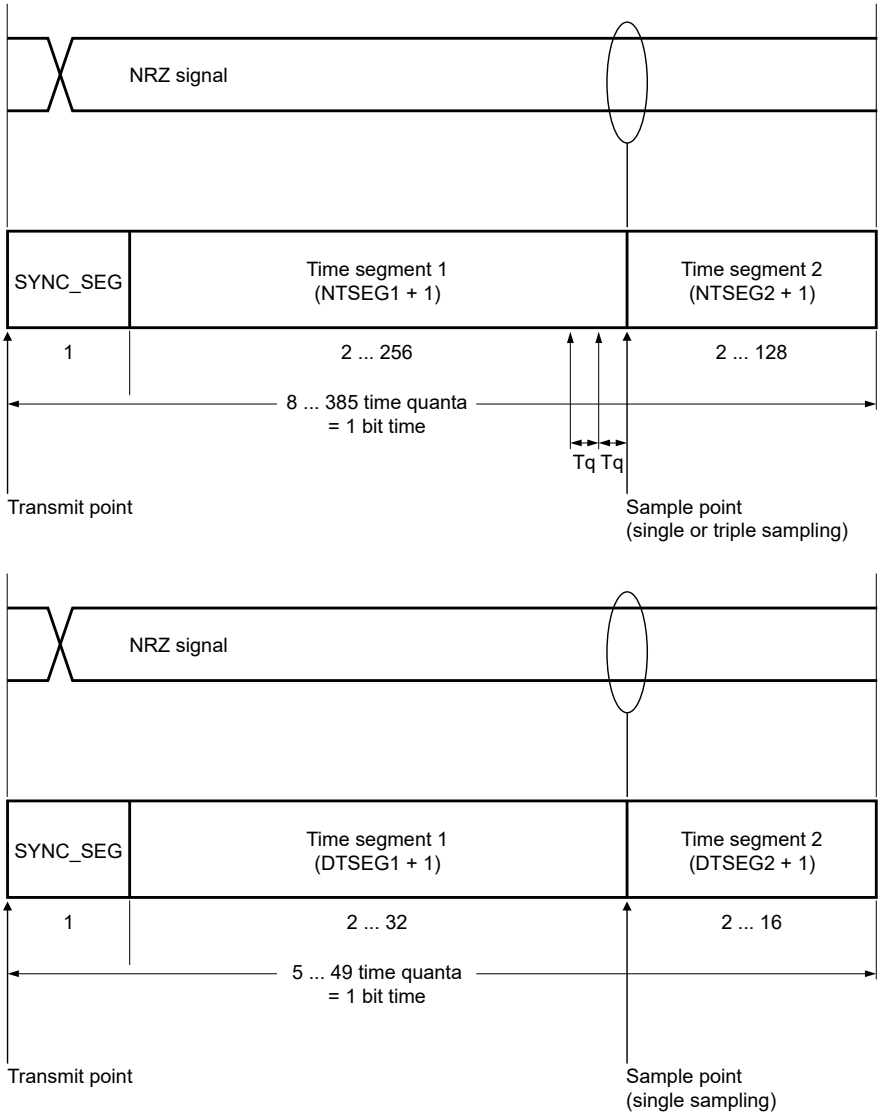


Figure 401. Segments within the bit timing (example using ENCBT and EDCBT bit timing variables)

The three bit time segments are:

- SYNC_SEG—this segment has a fixed length of one time quantum. Signal edges are expected to occur within this section.
- Time Segment 1—this segment includes the propagation segment and the phase segment 1 of the CAN standard.

  It can be programmed by configuring CTRL1[PROPSEG] and CTRL1[PSEG1] so that the sum (plus 2) is 2–16 time quanta. When CBT[BTF] = 1, FlexCAN uses CBT[EPROPSEG] and CBT[EPSEG1] so that the sum (plus 2) is 2–96 time quanta. For messages in CAN FD format with the BRS = 1, FlexCAN uses FDCBT[FPROPSEG] and FDCBT[FPSEG1] so that the sum (plus 1) is 2–39 time quanta.

  If CTRL2[BTE] = 1, FlexCAN uses ENCBT[NTSEG1] to configure time segment 1 to 2–256 time quanta. For the data phase in CAN FD messages with BRS = 1, EDCBT[DTSEG1] must be used for configuring time segment 1 to 2–32 time quanta.
- Time Segment 2—this segment represents the phase segment 2 of the CAN standard.

  It can be programmed by configuring CTRL1[PSEG2] (plus 1) to be 2–8 time quanta. When CBT[BTF] = 1, FlexCAN uses CBT[EPSEG2] so that its value (plus 1) is 2–32 time quanta. For messages in CAN FD format with the BRS = 1, FlexCAN uses

FDCBT[FPSEG2] instead, so that its value (plus 1) is 2–8 time quanta. Time segment 2 cannot be smaller than the Information Processing Time (IPT), which is 2 time quanta in FlexCAN.

If CTRL2[BTE] = 1, FlexCAN uses ENCBT[NTSEG2] to configure time segment 2 to 2–128 time quanta. For the data phase in CAN FD messages with BRS = 1, EDCBT[DTSEG2] must configure time segment 2 to 2–16 time quanta.

NOTE

The bit time defined by the above time segments must not be smaller than five time quanta. For bit time calculations, use an Information Processing Time (IPT) of two, which is the value implemented in the FlexCAN module.

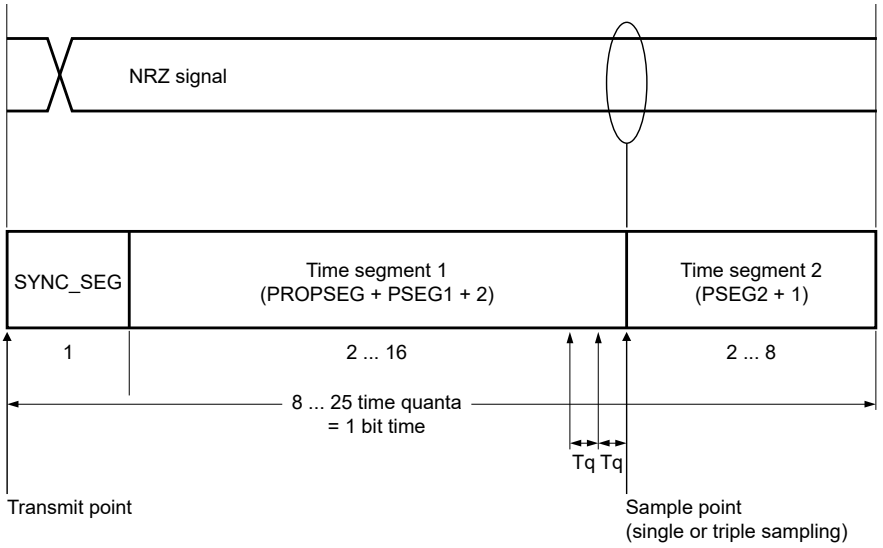


Figure 402. Segments within the bit time (example using CTRL1 bit timing variables for Classical CAN format)

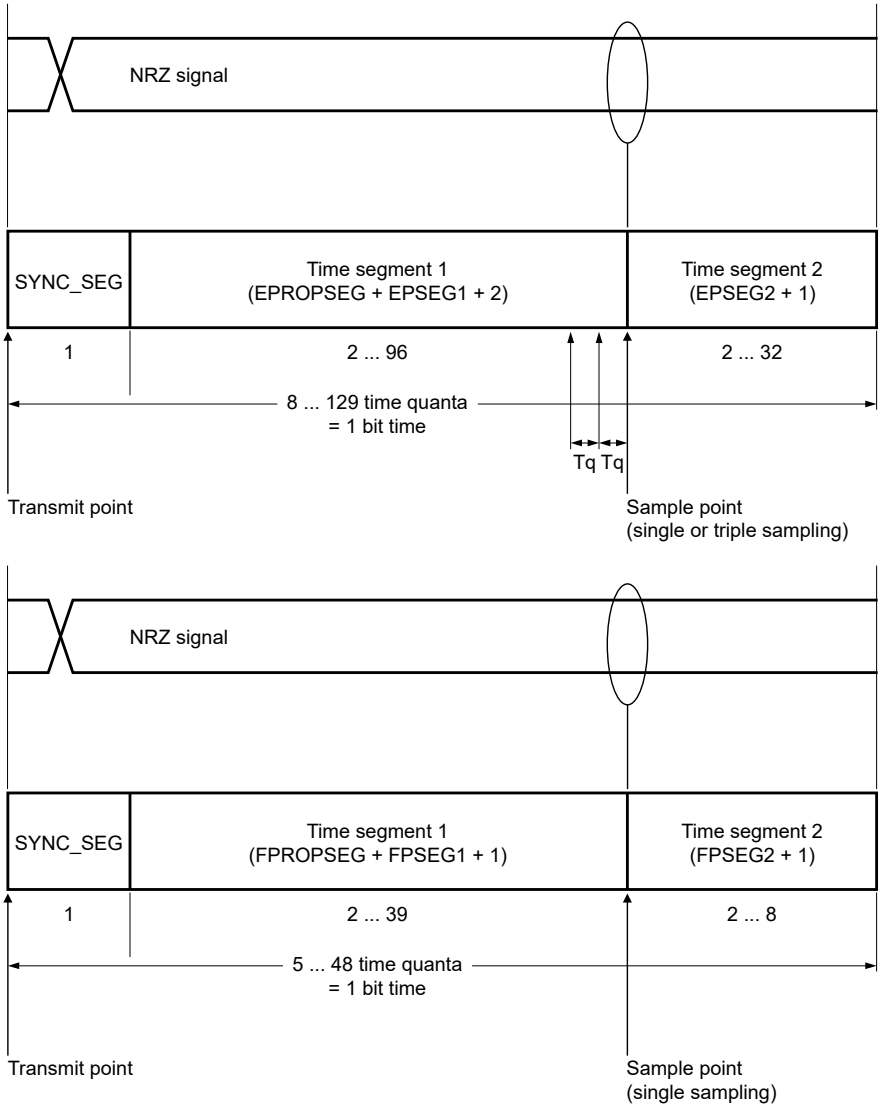


Figure 403. Segments within the bit time (example using CBT and FDCBT bit timing variables for CAN FD format)

Table 598. Time segment syntax

Syntax	Description
SYNC_SEG	Period during which the system expects transitions to occur on the bus
TSEG1	Period corresponding to the sum of PROPSEG and PSEG1
TSEG2	Period corresponding to the PSEG2 value
Transmit point	Point at which a node in Transmit mode transfers a new value to the CAN bus
Sample point	Point at which a node samples the bus. If the option of three samples per bit is selected, this point marks the position of the third sample.

Table 599 gives some examples of the CAN-compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 599. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Resynchronization jump width
5 to 10	2	1 to 2
4 to 11	3	1 to 3
5 to 12	4	1 to 4
6 to 13	5	1 to 4
7 to 14	6	1 to 4
8 to 15	7	1 to 4
9 to 16	8	1 to 4

NOTE

You must ensure the bit time settings comply with the CAN Protocol standard (ISO 11898-1:2015).

68.3.11.7.3 Calculating peripheral clocks

A CAN bit can be used as a measure of duration (for example, estimating the occurrence of a CAN bit event in a message). When a CAN bit is used in this way, the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$NumClkBit = \frac{f_{SYS}}{f_{CANCLK}} \times (PRESDIV + 1) \times (PROPSEG + PSEG1 + PSEG2 + 4)$$

Equation 32. Number of peripheral clocks per CAN bit when CTRL2[BTE] = 0

Or, if CTRL2[BTE] = 1:

$$NumClkBit = \frac{f_{SYS}}{f_{CANCLK}} \times (ENPRESDIV + 1) \times (NTSEG1 + NTSEG2 + 3)$$

Equation 33. Number of peripheral clocks per CAN bit when CTRL2[BTE] = 1

Where:

- NumClkBit is the number of peripheral clocks in one CAN bit.
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure 400), in Hz.
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz.
- PSEG1 is the value of CTRL1[PSEG1].
- PSEG2 is the value of CTRL1[PSEG2].
- PROPSEG is the value of CTRL1[PROPSEG].
- PRESDIV is the value in CTRL1[PRESDIV].
- ENPRESDIV is the value of EPRS[ENPRESDIV].
- NTSEG1 is the value of ENCBT[NTSEG1].
- NTSEG2 is the value of ENCBT[NTSEG2].

The formula above is also applicable to the alternative CAN bit timing variables described in:

- CAN Bit Timing (CBT)

- Enhanced Nominal CAN Bit Timing (ENCBT)
- CAN FD Bit Timing (FDCBT)
- Enhanced Nominal CAN Bit Timing (ENCBT)

For example, 180 CAN bits = (180 × NumClkBit) peripheral clock periods.

68.3.11.8 Arbitration and matching timing

During normal reception and transmission, the matching, arbitration, move-in, and move-out processes are executed during certain time windows inside the CAN frame. These windows are shown in the following figures.

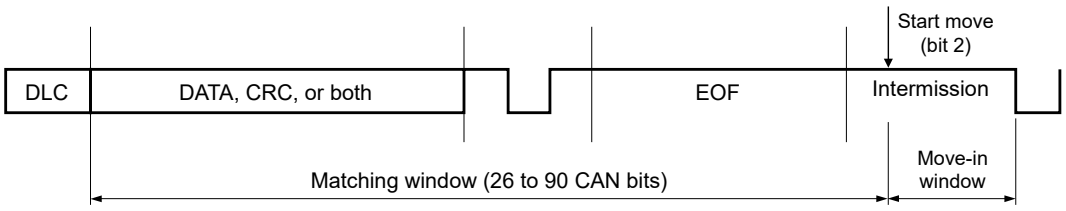


Figure 404. Matching and move-in time windows

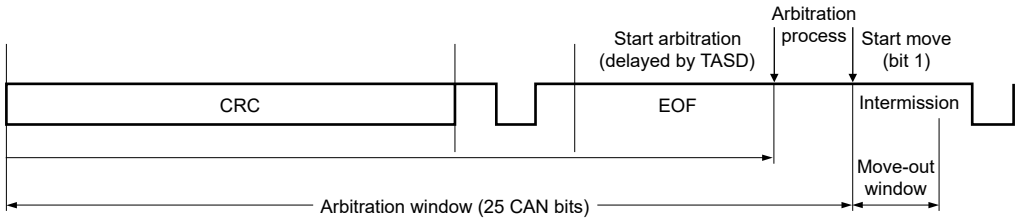


Figure 405. Arbitration and move-out time windows

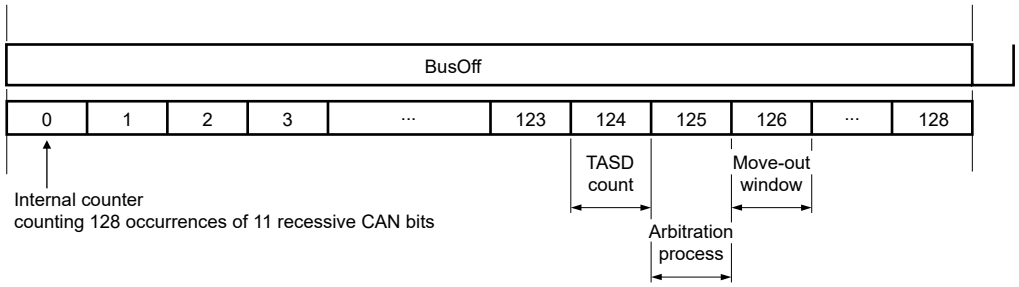


Figure 406. Arbitration at the end of bus off and move-out time windows

NOTE

In these figures, the matching and arbitration timing do not consider delays caused by concurrent memory access due to the CPU or other internal FlexCAN subblocks.

68.3.11.9 TX arbitration start delay

TX Arbitration Start Delay (CTRL2[TASD]) indicates the number of CAN bits that FlexCAN uses to delay the TX arbitration process starting point from the first bit of the CRC field of the current frame. This variable can be written only in Freeze mode; FlexCAN blocks it in other modes.

The ability of the CPU to reconfigure message buffers for transmission after the end of the internal arbitration process impacts transmission performance. In the arbitration process, FlexCAN finds the winner MB for transmission (see Arbitration process). If

the arbitration ends too early (before the first bit of the Intermission field) the CPU may reconfigure some TX message buffers. It is possible that the winning message buffer is no longer the best candidate to be transmitted.

TASD can optimize the transmission performance by defining the arbitration start point, as shown in Figure 407, based on factors such as:

- Peripheral-to-oscillator clock ratio
- CAN bit timing variables that determine the CAN bit rate
- Number of message buffers in use by the matching and arbitration processes

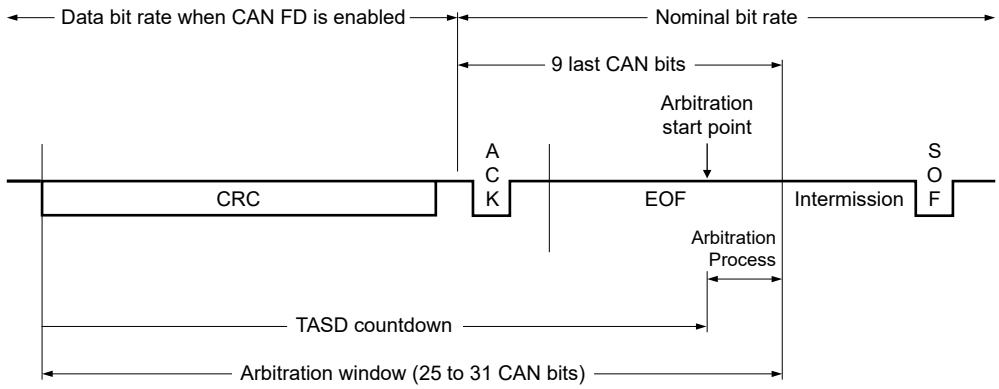


Figure 407. Optimal TX arbitration start point

The duration of an arbitration process, in terms of CAN bits, is:

- Directly proportional to the number of available message buffers
- Directly proportional to the CAN bit rate
- Inversely proportional to the peripheral clock frequency

The optimal arbitration timing occurs when the last message buffer is scanned immediately before the first bit of the Intermission field of a CAN frame. For instance, if the following are true:

- There are few message buffers.
- The peripheral-to-oscillator clock ratio is high.
- The CAN baud rate is low.

Then the arbitration can be placed closer to the end of the frame, adding more delay to its starting point, and vice versa.

If CTRL2[TASD] = 0, the arbitration start is not delayed, and more time is reserved for arbitration. Alternatively, if CTRL2[TASD] is close to 24, the CPU can configure a TX message buffer later, and less time is reserved for arbitration. If too little time is reserved for arbitration, FlexCAN may not be able to find a winner MB in time. The transmitted arbitration winner may not have the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

For CAN FD frames and (MAXMB + 1) ≤ NMB_{END}

$$TASD = 31 - \frac{2 * (MAXMB + 1) + 4}{CPCB_N}$$

For CAN FD frames and (MAXMB + 1) > NMB_{END}

$$TASD = 22 - \frac{2 * (MAXMB + 1) - NMB_{END}}{CPCB_F}$$

For non-FD frames

$$TASD = 25 - \frac{2 * (MAXMB + 1) + 4}{CPCB}$$

Equation 34. Optimal value for TASD

Where:

$$NMB_{END} = \frac{(9 \times CPCB_N) - 4}{2}$$

$$BITRATE_N = \left(\frac{f_{CANCLK}}{[1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] \times (EPRESDIV + 1)} \right)$$

$$BITRATE_F = \left(\frac{f_{CANCLK}}{[1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] \times (FPRESDIV + 1)} \right)$$

$$CPCB_N = \frac{f_{SYS}}{BITRATE_N}$$

$$CPCB_F = \frac{f_{SYS}}{BITRATE_F}$$

$$CPCB = CPCB_N$$

Equation 35. Variables used in TASD calculation

- MAXMB is the value in MCR[MAXMB].
- NMB_{END} is the number of message buffers that the arbitration process can scan during the last nine CAN bits at the end of a frame. (See Equation 35.)
- $BITRATE_N$ is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables.
- $BITRATE_F$ is the CAN bit rate in bits per second calculated by the data CAN bit time variables.
- $CPCB_N$ is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames.
- $CPCB_F$ is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames.
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames.
- f_{CANCLK} is the oscillator clock, in Hz.
- f_{SYS} is the peripheral clock, in Hz.
- EPSEG1 is the value in CBT[EPSEG1] (CTRL1[PSEG1] can also be used).
- EPSEG2 is the value in CBT[EPSEG2] (CTRL1[PSEG2] can also be used).
- EPROPSEG is the value in CBT[EPROPSEG] (CTRL1[PROPSEG] can also be used).
- EPRESDIV is the value in CBT[EPRESDIV] (CTRL1[PRESDIV] can also be used).
- FPSEG1 is the value in FDCBT[FPSEG1].

- FPSEG2 is the value in FDCBT[FPSEG2].
- FPROPSEG is the value in FDCBT[FPROPSEG].
- FPRESDIV is the value in FDCBT[FPRESDIV].
- NTSEG1 is the value in ENCBT[NTSEG1].
- NTSEG2 is the value in ENCBT[NTSEG2].
- ENPRESDIV is the value in EPRS[ENPRESDIV].
- DTSEG1 is the value in EDCBT[DTSEG1].
- DTSEG2 is the value in EDCBT[DTSEG2].
- EDPRESDIV is the value in EPRS[EDPRESDIV].

If CTRL2[BTE] = 1, then:

$$BITRATE_N = \frac{f_{CANCLK}}{[1 + (NTSEG1 + 1) + (NTSEG2 + 1)] \times (ENPRESDIV + 1)}$$

Equation 36. Nominal baud rate when CTRL2[BTE] = 1

$$BITRATE_F = \frac{f_{CANCLK}}{[1 + (DTSEG1 + 1) + (DTSEG2 + 1)] \times (EDPRESDIV + 1)}$$

Equation 37. Fast baud rate when CTRL2[BTE] = 1

See also Protocol timing for more details.

68.3.11.9.1 TASD configuration examples

The following tables show the TASD value calculated for some configuration cases.

Case 1:

- Clock ratio = 2:1 (for example, peripheral clock 80 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 600. TASD values in Case 1

Number of message buffers	TASD value	Maximum bit rate in data phase (MBd)
16	24	Invalid
32	24	8.0

Case 2:

- Clock ratio = 1:1 (for example, peripheral clock 40 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 601. TASD values in Case 2

Number of message buffers	TASD value	Maximum bit rate in data phase (MBd)
16	24	Invalid
32	23	6.67

Case 3:

- Clock ratio = 2:1 (for example, peripheral clock 40 MHz and oscillator clock 20 MHz)

- Bit rate in arbitration phase = 1 Mbaud

Table 602. TASD values in Case 3

Number of message buffers	TASD value	Maximum bit rate in data phase (MBd)
16	24	Invalid
32	23	4.0

68.3.12 Clocks

The following table describes the clock sources for FlexCAN. See the chip clocking chapter for clock setting, configuration, and gating information.

Table 603. FlexCAN clocks

Clock name	Description
MODULE_CLK (system_clk)	Peripheral clock
MODULE_CLK_CHI (host_clock)	Control Host Interface (CHI) clock
MODULE_CLK_PE (protocol_engine_clock)	Protocol Engine (PE) clock
MODULE_CLK_PE_NOGATE (protocol_engine_clock_nogate)	Protocol Engine clock (no gating)
MODULE_CLK_S (system_clock_nogate)	Peripheral access clock

68.3.12.1 Clock domains and restrictions

FlexCAN has two clock domains asynchronous to each other:

- The bus domain feeds the Control Host Interface (CHI) submodule.
- The oscillator domain feeds the CAN Protocol Engine (PE) submodule.

When the two domains are connected to clocks with different frequencies or phases, the frequency relationship between the two clock domains is restricted. In asynchronous operation, the bus domain clock frequency must always be greater than the oscillator domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

When performing matching and arbitration, FlexCAN must scan the whole message buffer memory during the time slot of one CAN frame, comprised of a number of CAN bits. To provide sufficient time for the scan, observe the following requirements:

- The peripheral clock frequency cannot be less than the oscillator clock frequency.
- There must be a minimum number of peripheral clocks per CAN bit, as specified in Table 604.

Table 604. Minimum number of peripheral clocks per CAN bit for Classical CAN format

Number of message buffers	Value of MCR[RFEN]	Value of ERFCR[ERFEN]	Minimum number of peripheral clocks per CAN bit
16	0	0	16
32	0	0	16

Table continues on the next page...

Table 604. Minimum number of peripheral clocks per CAN bit for Classical CAN format (continued)

Number of message buffers	Value of MCR[RFEN]	Value of ERFCR[ERFEN]	Minimum number of peripheral clocks per CAN bit
16	1	0	16
32	1	0	17
16	0	1	16
32	0	1	19

For classical frame format, the minimum number of peripheral clocks per CAN bit specified in Table 604 determines the minimum peripheral clock frequency for a given number of message buffers and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit. This number can be defined by adjusting one or more of the bit timing values contained in:

- Control 1 (CTRL1)
- CAN Bit Timing (CBT)
- Enhanced Nominal CAN Bit Timing (ENCBT)

The time quantum (Tq) is defined in Protocol timing. The minimum number of time quanta per CAN bit must be eight, so the oscillator clock frequency should be at least eight times the CAN bit rate.

68.3.12.1.1 Clock restrictions for CAN FD

For CAN FD frame format, some constraints must be satisfied. The equation below calculates the number of peripheral clocks per CAN bit in nominal bit rate (NumClkNomBit).

$$NumClkNomBit = \frac{f_{SYS}}{f_{CANCLK}} \times (PRESDIV + 1) \times (PROPSEG + PSEG1 + PSEG2 + 4)$$

$$= \frac{f_{SYS}}{NomBitRate}$$

Equation 38. Number of peripheral clocks per nominal CAN bit

Where PRESDIV, PSEG1, and PSEG2 are CAN bit time values in Control 1 (CTRL1). Alternatively, EPRESDIV, EPSEG1, and EPSEG2 values in CAN Bit Timing (CBT) or the values of EPRS[ENPRESDIV], ENCBT[NTSEG1], and ENCBT[NTSEG2] can be used instead. NumClkNomBit can also be calculated as a function of the expected nominal bit rate used in the arbitration phase (NomBitRate), as shown in the equation above.

The number of CAN bits in the data phase of an FD frame with BRS = 1 (fast CAN bits) depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) can be determined in Table 605. Having fewer data bytes means having fewer fast CAN bits. It also means that less time is available for FlexCAN to scan the whole message buffer memory during the internal matching and arbitration processes.

Table 605. Number of fast CAN bits in a CAN FD frame

Minimum number of data bytes	DLC field	NumOfFastBits
0	0h	21
1	1h	29
2	2h	37
3	3h	45
4	4h	53

Table continues on the next page...

Table 605. Number of fast CAN bits in a CAN FD frame (continued)

Minimum number of data bytes	DLC field	NumOfFastBits
5	5h	61
6	6h	69
7	7h	77
8	8h	85
12	9h	117
16	Ah	149
20	Bh	186
24	Ch	218
32	Dh	282
48	Eh	410
64	Fh	538

The critical part of a CAN FD frame is during the data phase, where the CAN bit rate is faster than in the arbitration phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated to guarantee that enough time is available for FlexCAN to scan the message buffer memory during reception and transmission. The equation below calculates this constraint.

$$MinNumClkFastBit_A = \frac{(8.5 \times MaxNumOfMb) + [ERFEN \times (2 \times NFE + 4)] + 64 - (9 \times NumClkNomBit)}{NumOfFastBits}$$

Equation 39. Minimum number of peripheral clocks per fast CAN bit for FlexCAN scan process

Where MaxNumOfMb is the maximum number of available message buffers defined in MCR[MAXMB]. NFE and ERFEN are the fields defined in Enhanced RX FIFO Control (ERFCR).

The clock-domain-crossing circuit between the CHI and PE subblocks also imposes a minimum number of peripheral clocks per fast CAN bit. This minimum is required for the handshake mechanism to work properly without losing status information through the interface, as shown in the equation below.

$$MinNumClkFastBit_B = 3 \times \left(1 + \frac{f_{SYS}}{f_{CANCLK}}\right)$$

Equation 40. Minimum number of peripheral clocks per fast CAN bit for FlexCAN clock domain interface

Therefore, the larger of the two values calculated above determines the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit).

$$MinNumClkFastBit = Maximum\ (\ MinNumClkFastBitA, MinNumClkFastBitB)$$

Equation 41. Minimum number of peripheral clocks per fast CAN bit

Then, the maximum CAN bit rate in the data phase of CAN FD frames (DataBitRateMAX) can be calculated as below.

$$DataBitrate_{MAX} = \frac{f_{CANCLK}}{ROUNDUP\left(\frac{MinNumClkFastBit \times f_{CANCLK}}{f_{SYS}}\right)}$$

Equation 42. Maximum achievable baud rate for data phase

These factors affect the maximum data bit rate attainable by FlexCAN in CAN FD mode:

- The peripheral and oscillator clock frequencies
- The maximum number of message buffers
- The expected nominal bit rate

Also, the data bit rate depends on the minimum payload size of FD frames used in a given application.

To illustrate how the configuration of FlexCAN variables affects the CAN FD bit rate, consider this application example:

- The peripheral clock frequency is set to 50 MHz
- The oscillator clock frequency is set to 40 MHz

1. Considering the nominal bit rate as 1 Mbit/s, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as below.

$$NumClkNomBit = \frac{50 \times 10^6}{1 \times 10^6} = 50$$

Equation 43. Calculation example for number of peripheral clocks per nominal CAN bit

2. The number of fast CAN bits (NumOfFastBits) is determined in Table 605. For example, if the minimum payload in FD frames is 8 bytes, there are 85 CAN bits in the data phase.
3. Assuming the maximum number of message buffers is 96, and Enhanced RX FIFO is disabled, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated.

$$MinNumClkFastBit_A = \frac{(8.5 \times 96) + 64 - (9 \times 50)}{85} = 5.06$$

Equation 44. Calculation example for number of peripheral clocks per fast CAN bit for FlexCAN scan process

$$MinNumClkFastBit_B = 3 \times \left(1 + \frac{50}{40}\right) = 6.75$$

Equation 45. Calculation example for number of peripheral clocks per fast CAN bit for FlexCAN clock domain interface

$$MinNumClkFastBit = Maximum\ (\ 5.06, 6.75\ ) = 6.75$$

Equation 46. Calculation example for number of peripheral clocks per fast CAN bit

4. The maximum CAN bit rate in the data phase can finally be found.

$$DataBitRate_{MAX} = \frac{40 \times 10^6}{ROUNDUP\left(\frac{6.75 \times 40 \times 10^6}{50 \times 10^6}\right)} = 6.667\ Mbps$$

Equation 47. Calculation example for maximum achievable baud rate

Even though the oscillator clock frequency (40 MHz) is adequate to generate a data rate of 8 Mbit/s in CAN FD mode, the specific FlexCAN configuration limits this rate to 6.667 Mbit/s. This limitation is mainly due to the low peripheral clock frequency that imposes the MinNumClkFastBitB bound.

Table 606 shows the maximum data rate for CAN FD with Enhanced RX FIFO disabled according to clock frequencies, payload size, and number of available message buffers. For some cases, if the number of available message buffers is reduced, FlexCAN can then achieve a data rate up to 8 Mbit/s.

Table 606. Maximum CAN bit rate in data phase on CAN FD frames with Enhanced RX FIFO disabled

Peripheral clock frequency (MHz)	Payload size	Number of available message buffers	Maximum data rate (Mbit/s)
40	8	94	6.667
40	8	114	>5.0
40	12	>117	6.667
40	12	128	5.714
50	12–64	128	6.667
60	8	126	8.0
60	12	128	8.0
67	6	128	8.0
80	3	128	8.0
100	0	128	8.0

68.3.13 Reset

You can reset FlexCAN in the following ways:

- Chip-level hard reset, which resets all memory-mapped registers asynchronously.
- Soft reset, in one of two ways:
  - MCR[SOFTRST], which resets some of the memory-mapped registers synchronously. To see which registers soft reset affects, see Table 609.
  - Chip-level soft reset, which has the same effect as MCR[SOFTRST].

Soft reset is synchronous and must follow an internal request-and-acknowledge procedure across clock domains. Therefore, it may take some time to propagate its effects fully. MCR[SOFTRST] remains 1 when soft reset is pending, so software can poll this field to identify when the reset has completed. Soft reset cannot be applied when clocks are shut down in a low-power mode. The low-power mode should be exited and the clocks resumed before applying soft reset.

When the module is enabled (MCR[MDIS] becomes 0), FlexCAN automatically enters Freeze mode. In Freeze mode:

1. FlexCAN is unsynchronized to the CAN bus.
2. MCR[HALT] and MCR[FRZ] become 1.
3. The internal state machines are disabled.
4. MCR[FRZACK] and MCR[NOTRDY] become 1.

The TX pin is in the recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Reset does not affect the message buffers and the RX Individual Mask registers, so they are not automatically initialized.

68.3.14 Interrupts

FlexCAN has many interrupt sources:

- Interrupts due to message buffers
- Interrupts due to interrupts combined via an OR operator from:
  - Message buffers
  - Bus Off
  - Bus Off Done
  - Error
  - Error Fast (errors detected in the data phase of CAN FD format messages with BRS = 1)
  - Wake Up
  - Wake Up Match
  - Wake Up Timeout
  - TX Warning
  - RX Warning

If its corresponding IMASK bit is 1, each message buffer can be an interrupt source. There is no distinction between TX and RX interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each buffer has an assigned flag bit in the IFLAG registers. When the corresponding buffer completes a successful transfer, the flag is set. When the CPU writes 1 to it, the flag is cleared (unless another interrupt is generated at the same time).

NOTE

The CPU must clear only the bit causing the current interrupt. For this reason, do not use bit manipulation instructions (BSET) to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt handler.

If the Legacy RX FIFO is enabled (MCR[RFEN] = 1) and DMA is disabled (MCR[DMA] = 0), the interrupts corresponding to message buffers 0–7 have different meanings.

- Bit 7 of Interrupt Flags 1 (IFLAG1) becomes the Legacy FIFO Overflow flag
- Bit 6 becomes the Legacy FIFO Warning flag
- Bit 5 becomes the Frames Available in Legacy FIFO flag
- Bits 4–0 are unused.

See Interrupt Flags 1 (IFLAG1) for more information.

If both Legacy RX FIFO and DMA are enabled (MCR[RFEN] = 1 and MCR[DMA] = 1), FlexCAN does not generate any Legacy FIFO interrupt. Bit 5 of IFLAG1 still indicates Frames Available in Legacy FIFO and generates a DMA request. Bits 7, 6, and 4–0 are unused.

CAUTION

Legacy FIFO cannot be enabled when CAN FD is enabled.

When multiple message buffer interrupt sources are combined via an OR operator into a single interrupt, the interrupt is generated when any associated message buffer (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which message buffer or FIFO source caused the interrupt.

These interrupt sources generate interrupts like the message buffer interrupt sources, and can be read from Error and Status 1 (ESR1):

- Bus Off
- Bus Off Done
- Error
- Error Fast

- Wake Up
- TX Warning
- RX Warning

The Bus Off, Error, TX Warning, and RX Warning interrupt masks are located in Control 1 (CTRL1); the wake-up interrupt mask is located in Module Configuration (MCR).

The interrupt sources for Pretended Networking (Wake-up by Match Flag and Wake-up by Timeout Flag) can be read in Pretended Networking Wake-Up Match (WU_MTC). The respective interrupt mask bits are located in Pretended Networking Control 1 (CTRL1_PN).

68.3.15 Bus interface

CPU access to FlexCAN registers is subject to the following rules:

- Read and write access to implemented reserved address space results in an access error.
- Write access to positions whose bits are all currently read-only results in an access error. If at least one of the bits is not read-only, no access error is issued. Write permission to specific positions or some of their bits can change depending on the mode of operation or transitory state. See register and field descriptions for details.
- Read and write access to unimplemented address space results in an access error.
- Read and write access to RAM-located positions during Low-Power mode results in an access error.
- The RXIMR memory region can be considered as general-purpose memory and available for access via these methods:
  - If you write 0 to MCR[IRMQ], the individual masks (RXIMR) are disabled. In this case, the RXIMR memory region is considered general-purpose memory.
  - If MCR[MAXMB] is programmed with a value smaller than the available number of message buffers, the unused memory space can be used as general-purpose RAM space. Reserved words within RAM cannot be used.
    For example, suppose the RAM in FlexCAN can support up to 16 message buffers, CTRL2[RFFN] = 0h, and MCR[MAXMB] = 0.
    - In this case, the maximum number of message buffers becomes one.
    - The RAM starts at 0080h, and the space 0080h–008Fh is used by the one message buffer.
    - The memory space 0090h–017Fh is available.
    - The space 0180h–087Fh is reserved.
    - The space 0880h–0883h is used by the one individual mask and the available memory in the mask register space is 0884h–08BFh.
    - In the space from 08C0h–09DFh, there are reserved words for internal use which cannot be used as general-purpose RAM.

    As a rule, free memory space for general purpose depends only on MCR[MAXMB].
  - If MCR[FDEN] = 1, general-purpose memory can be used only outside Freeze mode.

Table 607. Access permissions

Modes of operation	Normal	Freeze	Low-power
MCR	Bus error	Bus error	Bus error
CTRL1	Read and write	Read and write	Read and write
TIMER	Read and write	Read and write	Read and write
TCR	Bus error	Bus error	Bus error

Table continues on the next page...

Table 607. Access permissions (continued)

Modes of operation	Normal	Freeze	Low-power
RXMGMASK[1]	Bus error for write operation	Read and write	Bus error for write operation
RX14MASK[1]	Bus error for write operation	Read and write	Bus error for write operation
RX15MASK[1]	Bus error for write operation	Read and write	Bus error for write operation
ECR	Bus error for write operation	Read and write	Bus error for write operation
ESR1	Read and write	Read and write	Read and write
IMASK1	Read and write	Read and write	Read and write
IFLAG1	Read and write	Read and write	Read and write
CTRL2	Read and write	Read and write	Read and write
ESR2	Read and write	Read and write	Read and write
CRCR	Bus error for write operation	Bus error for write operation	Bus error for write operation
RXFGMASK[1]	Bus error for write operation	Read and write	Bus error for write operation
RXFIR[1]	Bus error for write operation	Bus error for write operation	Bus error for write operation
CBT	Bus error for write operation	Read and write	Bus error for write operation
DBG1	Bus error for write operation	Bus error for write operation	Bus error for write operation
DBG2	Bus error for write operation	Bus error for write operation	Bus error for write operation
MB[1] [2]	Read and write	Read and write	Read and write
Legacy FIFO header[1]	Read and write	Read and write	Read and write
Legacy FIFO reserved space[1]	Bus error	Bus error	Bus error
Legacy FIFO filters[1]	Read and write	Read and write	Read and write
RXIMR[1]	Bus error	Read and write	Bus error
CTRL1_PN	Read and write [3]	Read and write	Read and write [3]
CTRL2_PN	Read and write [3]	Read and write	Read and write [3]
WU_MTC	Read and write	Read and write	Read and write
FLT_ID1	Read and write [3]	Read and write	Read and write [3]
FLT_DLC	Read and write [3]	Read and write	Read and write [3]

Table continues on the next page...

Table 607. Access permissions (continued)

Modes of operation	Normal	Freeze	Low-power
PL1_LO	Read and write [3]	Read and write	Read and write [3]
PL1_HI	Read and write [3]	Read and write	Read and write [3]
FLT_ID2_IDMASK	Read and write [3]	Read and write	Read and write [3]
PL2_PLMASK_LO	Read and write [3]	Read and write	Read and write [3]
PL2_PLMASK_HI	Read and write [3]	Read and write	Read and write [3]
WMB0	Read and write [3]	Read and write [3]	Read and write [3]
WMB1	Read and write [3]	Read and write [3]	Read and write [3]
WMB2	Read and write [3]	Read and write [3]	Read and write [3]
WMB3	Read and write [3]	Read and write [3]	Read and write [3]
EPRS	Bus error for write operation	Read and write	Bus error for write operation
ENCBT	Bus error for write operation	Read and write	Bus error for write operation
EDCBT	Bus error for write operation	Read and write	Bus error for write operation
ETDC	Bus error for write operation	Read and write	Bus error for write operation
FDCTRL	Read and write	Read and write	Read and write
FDCBT	Read and write [3]	Read and write	Read and write [3]
FDCRC	Bus error for write operation	Bus error for write operation	Bus error for write operation
ERFCR	Read and write [3]	Read and write	Read and write [3]
ERFIER	Read and write	Read and write	Read and write
ERFSR	Read and write	Read and write	Read and write
Enhanced Rx FIFO header[4]	Bus error for write operation	Bus error for write operation	Bus error for write operation
Enhanced Rx FIFO reserved space[4]	Bus error	Bus error	Bus error
ERFFEL	Bus error for write operation	Read and write	Bus error for write operation
General purpose RAM[1]	Read and write	Read and write	Read and write
Reserved space (used)[1]	Bus error	Bus error	Bus error
Reserved space (empty)[1]	Bus error	Bus error	Bus error

1. Access in low power is only possible if RAM_clk and MODULE_CLK are enabled.
2. If MCR[RFEN] = 1, see Legacy FIFO access rules below.
3. Write operation has no effect.
4. If MCR[RFEN] = 1, do not access Enhanced RX FIFO.

68.4 External signal descriptions

FlexCAN has two I/O signals connected to the external chip pins. These signals are summarized in Table 608 and described in the following subsections.

Table 608. FlexCAN signal descriptions

Signal	Description	I/O
CAN RX	CAN receive pin	Input
CAN TX	CAN transmit pin	Output

68.4.1 CAN RX

This pin is the receive pin from the CAN bus transceiver. Logic level 0 represents its dominant state. Logic level 1 represents its recessive state.

68.4.2 CAN TX

This pin is the transmit pin to the CAN bus transceiver. Logic level 0 represents its dominant state. Logic level 1 represents its recessive state.

68.5 Initialization and application information

68.5.1 FlexCAN initialization sequence

For any configuration change or initialization, you must put FlexCAN into Freeze mode (see Freeze mode). The module must be initialized after every reset. FlexCAN memory must be initialized before switching to functional mode.

The following is a generic initialization sequence applicable to FlexCAN:

1. Initialize Module Configuration (MCR).
    a. Enable the individual filtering per message buffer and reception queue features by writing 1 to MCR[IRMQ].
    b. Enable the warning interrupts by writing 1 to MCR[WRNEN].
    c. If required, disable frame self-reception by writing 1 to MCR[SRXDIS].
    d. Enable the Legacy RX FIFO by writing 1 to MCR[RFEN] or enable the Enhanced RX FIFO by writing 1 to ERFCR[ERFEN].
    e. If Legacy RX FIFO or Enhanced RX FIFO is enabled and DMA is required, write 1 to MCR[DMA].
    f. If Pretended Networking mode is required, write 1 to MCR[PNET_EN].
    g. Enable the abort mechanism by writing 1 to MCR[AEN].
    h. Enable the local priority feature by writing 1 to MCR[LPRIOEN].
2. Initialize Control 1 (CTRL1) and CAN FD Bit Timing (FDCBT). Optionally initialize CAN Bit Timing (CBT).
    a. Determine the bit timing parameters: CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW].
    b. Optionally determine the bit timing parameters: CBT[EPROPSEG], CBT[EPSEG1], CBT[EPSEG2], and CBT[ERJW].
    c. Determine the CAN FD bit timing parameters: FDCBT[FPROPSEG], FDCBT[FPSEG1], FDCBT[FPSEG2], and FDCBT[FRJW].
    d. Determine the bit rate by programming CTRL1[PRESDIV] and optionally programming CBT[EPRESDIV].
    e. Determine the CAN FD bit rate by programming FDCBT[FPRESDIV].
    f. Determine the internal arbitration mode by programming CTRL1[LBUF].

3. Initialize the message buffers. (See Message buffer structure for message buffer details.)
    a. The control and status word of all message buffers must be initialized.
    b. If RX FIFO is enabled, the ID filter table must be initialized.
    c. Other entries in each message buffer should be initialized as required.
4. Initialize Receive Individual Mask (RXIMR0 - RXIMR31).
5. Write 1 to required interrupt mask bits in:
    - IMASK registers (for all message buffer interrupts)
    - Module Configuration (MCR) (for wake-up interrupt)
    - Control 1 (CTRL1) and Control 2 (CTRL2) (for Bus Off and Error interrupts)
6. If Pretended Networking mode is enabled, configure the necessary registers for selective wake-up.
7. Write 0 to MCR[HALT].

After the last step listed above, FlexCAN attempts to synchronize to the CAN bus.

68.6 Memory map and register definition

This section describes the registers and data structures in FlexCAN. The base address of the module depends on the particular memory map of the chip.

68.6.1 FlexCAN memory mapping

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address offset 0080h.

Each individual register is identified by its complete name and corresponding mnemonic.

NOTE

An invalid register access results in a bus error. Invalid accesses include reading a write-only register, writing a read-only register, and accessing an invalid address.

NOTE

To update the parity bits in memory properly, all FlexCAN memory must be initialized before reading registers which are implemented in memory. You must also initialize RX Message Buffers Global Mask (RXMGMASK), Receive 14 Mask (RX14MASK), Receive 15 Mask (RX15MASK) and Legacy RX FIFO Global Mask (RXFGMASK). MCR[RFEN] and ERFCR[ERFEN] must not be 1 during memory initialization.

Table 609. Register reset information

Register	Affected by hard reset	Affected by soft reset
Module Configuration (MCR)	Yes	Yes
Control 1 (CTRL1)	Yes	No
Free-Running Timer (TIMER)	Yes	Yes
RX Message buffers Global Mask (RXMGMASK)	No	No
RX Buffer 14 Mask (RX14MASK)	No	No

Table continues on the next page...

Table 609. Register reset information (continued)

Register	Affected by hard reset	Affected by soft reset
RX Buffer 15 Mask (RX15MASK)	No	No
Error Counter (ECR)	Yes	Yes
Error and Status 1 (ESR1)	Yes	Yes
Interrupt Masks 1 (IMASK1)	Yes	Yes
Interrupt Flags 1 (IFLAG1)	Yes	Yes
Control 2 (CTRL2)	Yes	No
Error and Status 2 (ESR2)	Yes	Yes
Cyclic Redundancy Check (CRCR)	Yes	Yes
RX FIFO Global Mask (RXFGMASK)	No	No
RX FIFO Information (RXFIR)	No	No
CAN Bit Timing (CBT)	Yes	No
Message buffers	No	No
RX Individual Masks	No	No
Pretended Networking Control 1 (CTRL1_PN)	Yes	Yes
Pretended Networking Control 2 (CTRL2_PN)	Yes	Yes
Pretended Networking Wake-Up Match (WU_MTC)	Yes	Yes
Pretended Networking ID Filter 1 (FLT_ID1)	Yes	Yes
Pretended Networking DLC Filter (FLT_DLC)	Yes	Yes
Pretended Networking Payload Low Filter 1 (PL1_LO)	Yes	Yes
Pretended Networking Payload High Filter 1 (PL1_HI)	Yes	Yes
Pretended Networking ID Filter 2 or ID Mask (FLT_ID2_IDMASK)	Yes	Yes
Pretended Networking Payload Low Filter 2 or Payload Low Mask (PL2_PLMASK_LO)	Yes	Yes
Pretended Networking Payload High Filter 2 or Payload High Mask (PL2_PLMASK_HI)	Yes	Yes
Pretended Networking Wake Up Message Buffer 0 (WMB0)	Yes	No
Pretended Networking Wake Up Message Buffer 1 (WMB1)	Yes	No

Table continues on the next page...

Table 609. Register reset information (continued)

Register	Affected by hard reset	Affected by soft reset
Pretended Networking Wake-Up Message Buffer 2 (WMB2)	Yes	No
Pretended Networking Wake-Up Message Buffer 3 (WMB3)	Yes	No
Enhanced CAN Bit Timing Prescalers (EPRS)	Yes	No
Enhanced Nominal CAN Bit Timing (ENCBT)	Yes	No
Enhanced Data Phase CAN bit Timing (EDCBT)	Yes	No
Enhanced Transceiver Delay Compensation (ETDC)	Yes	No
CAN FD Control (FDCTRL)	Yes	No
CAN FD Bit Timing (FDCBT)	Yes	No
CAN FD CRC (FDCRC)	Yes	Yes
Enhanced RX FIFO Control (ERFCR)	Yes	Yes
Enhanced RX FIFO Interrupt Enable (ERFIER)	Yes	Yes
Enhanced RX FIFO Status (ERFSR)	Yes	Yes
Enhanced RX FIFO	No	No
Enhanced RX FIFO Filter Element (ERFFEL)	No	No

FlexCAN can store CAN messages for transmission and reception using message buffers and RX FIFO structures.

68.6.2 CAN register descriptions

The table below shows the FlexCAN memory map.

The address range from offset 80h–27Fh allocates the thirty-two 128-bit message buffers. The memory maps for the message buffers are in FlexCAN message buffer memory map.

The address range from offset 2000h–2048h allocates the Enhanced RX FIFO output, and the address range from offset 204Ch–238Ch allocates the rest of Enhanced RX FIFO 11 elements. The memory map for the Enhanced RX FIFO is in Enhanced RX FIFO structure.

68.6.2.1 CAN memory map

CAN0 base address: 400D_4000h

CAN1 base address: 400D_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Configuration (MCR)	32	RW	D890_000Fh
4h	Control 1 (CTRL1)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
8h	Free-Running Timer (TIMER)	32	RW	0000_0000h
10h	RX Message Buffers Global Mask (RXMGMASK)	32	RW	See section
14h	Receive 14 Mask (RX14MASK)	32	RW	See section
18h	Receive 15 Mask (RX15MASK)	32	RW	See section
1Ch	Error Counter (ECR)	32	RW	0000_0000h
20h	Error and Status 1 (ESR1)	32	RW	0000_0000h
28h	Interrupt Masks 1 (IMASK1)	32	RW	0000_0000h
30h	Interrupt Flags 1 (IFLAG1)	32	RW	0000_0000h
34h	Control 2 (CTRL2)	32	RW	00A0_0000h
38h	Error and Status 2 (ESR2)	32	R	0000_0000h
44h	Cyclic Redundancy Check (CRCR)	32	R	0000_0000h
48h	Legacy RX FIFO Global Mask (RXFGMASK)	32	RW	See section
4Ch	Legacy RX FIFO Information (RXFIR)	32	R	See section
50h	CAN Bit Timing (CBT)	32	RW	0000_0000h
880h - 8FCh	Receive Individual Mask (RXIMR0 - RXIMR31)	32	RW	See section
B00h	Pretended Networking Control 1 (CTRL1_PN)	32	RW	0000_0100h
B04h	Pretended Networking Control 2 (CTRL2_PN)	32	RW	0000_0000h
B08h	Pretended Networking Wake-Up Match (WU_MTC)	32	RW	0000_0000h
B0Ch	Pretended Networking ID Filter 1 (FLT_ID1)	32	RW	0000_0000h
B10h	Pretended Networking Data Length Code (DLC) Filter (FLT_DLC)	32	RW	0000_0008h
B14h	Pretended Networking Payload Low Filter 1 (PL1_LO)	32	RW	0000_0000h
B18h	Pretended Networking Payload High Filter 1 (PL1_HI)	32	RW	0000_0000h
B1Ch	Pretended Networking ID Filter 2 or ID Mask (FLT_ID2_IDMASK)	32	RW	0000_0000h
B20h	Pretended Networking Payload Low Filter 2 and Payload Low Mask (PL2_PLMASK_LO)	32	RW	0000_0000h
B24h	Pretended Networking Payload High Filter 2 and Payload High Mask (PL2_PLMASK_HI)	32	RW	0000_0000h
B40h	Wake-Up Message Buffer (WMB0_CS)	32	R	0000_0000h
B44h	Wake-Up Message Buffer for ID (WMB0_ID)	32	R	0000_0000h
B48h	Wake-Up Message Buffer for Data 0–3 (WMB0_D03)	32	R	0000_0000h
B4Ch	Wake-Up Message Buffer Register Data 4–7 (WMB0_D47)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
B50h	Wake-Up Message Buffer (WMB1_CS)	32	R	0000_0000h
B54h	Wake-Up Message Buffer for ID (WMB1_ID)	32	R	0000_0000h
B58h	Wake-Up Message Buffer for Data 0–3 (WMB1_D03)	32	R	0000_0000h
B5Ch	Wake-Up Message Buffer Register Data 4–7 (WMB1_D47)	32	R	0000_0000h
B60h	Wake-Up Message Buffer (WMB2_CS)	32	R	0000_0000h
B64h	Wake-Up Message Buffer for ID (WMB2_ID)	32	R	0000_0000h
B68h	Wake-Up Message Buffer for Data 0–3 (WMB2_D03)	32	R	0000_0000h
B6Ch	Wake-Up Message Buffer Register Data 4–7 (WMB2_D47)	32	R	0000_0000h
B70h	Wake-Up Message Buffer (WMB3_CS)	32	R	0000_0000h
B74h	Wake-Up Message Buffer for ID (WMB3_ID)	32	R	0000_0000h
B78h	Wake-Up Message Buffer for Data 0–3 (WMB3_D03)	32	R	0000_0000h
B7Ch	Wake-Up Message Buffer Register Data 4–7 (WMB3_D47)	32	R	0000_0000h
BF0h	Enhanced CAN Bit Timing Prescalers (EPRS)	32	RW	0000_0000h
BF4h	Enhanced Nominal CAN Bit Timing (ENCBT)	32	RW	0000_0000h
BF8h	Enhanced Data Phase CAN Bit Timing (EDCBT)	32	RW	0000_0000h
BFCh	Enhanced Transceiver Delay Compensation (ETDC)	32	RW	0000_0000h
C00h	CAN FD Control (FDCTRL)	32	RW	8000_0100h
C04h	CAN FD Bit Timing (FDCBT)	32	RW	0000_0000h
C08h	CAN FD CRC (FDCRC)	32	R	0000_0000h
C0Ch	Enhanced RX FIFO Control (ERFCR)	32	RW	0000_0000h
C10h	Enhanced RX FIFO Interrupt Enable (ERFIER)	32	RW	0000_0000h
C14h	Enhanced RX FIFO Status (ERFSR)	32	RW	0000_0000h
3000h - 307Ch	Enhanced RX FIFO Filter Element (ERFFEL0 - ERFFEL31)	32	RW	See section

68.6.2.2 Module Configuration (MCR)

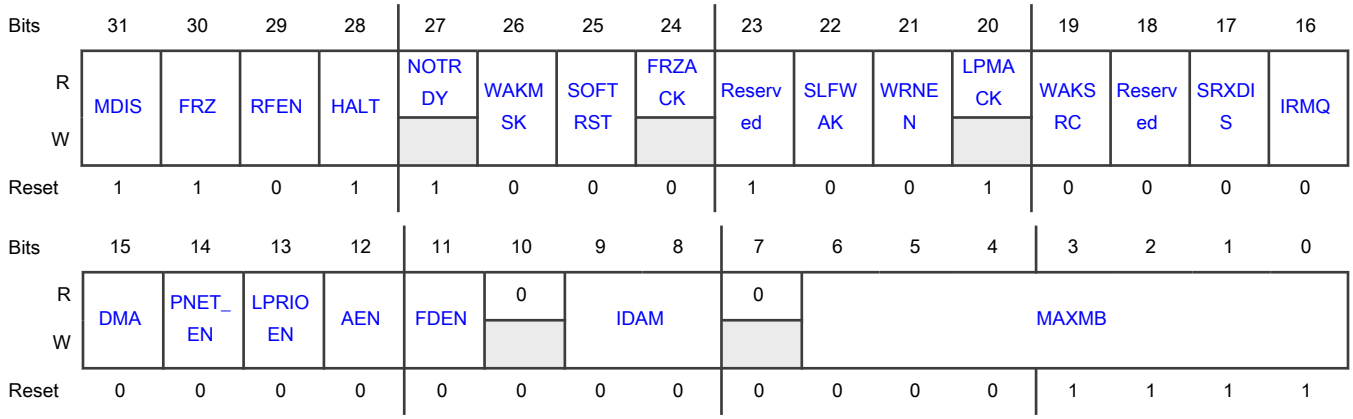
Offset

Register	Offset
MCR	0h

Function

Defines global system configurations, including the module operation modes and the maximum message buffer configuration.

Diagram



Fields

Field	Function
31 MDIS	Module Disable Disables FlexCAN. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Soft reset does not affect this field. 0b - Enable 1b - Disable
30 FRZ	Freeze Enable Specifies FlexCAN behavior when MCR[HALT] = 1 or when Debug mode is requested at chip level. When this field becomes 1, FlexCAN can enter Freeze mode. Writing 0 to this field causes FlexCAN to exit from Freeze mode. 0b - Disable 1b - Enable
29 RFEN	Legacy RX FIFO Enable Enables the Legacy RX FIFO feature. When this field is 1, message buffers 0–5 cannot be used for normal reception and transmission. The corresponding memory region (80h–DCh) is used by the FIFO engine and additional message buffers (up to 32, depending on CTRL2[RFFN]). These message buffers are used as Legacy RX FIFO ID filter table elements. This field also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 604. This field can be written in Freeze mode only; the module blocks it in other modes. **NOTE** When CAN FD operation is enabled (see MCR[FDEN]), you cannot write 1 to this field. **NOTE** This field must not be 1 if ERFCR[ERFEN] = 1. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
28 HALT	Halt FlexCAN Puts FlexCAN into Freeze mode. The CPU should write 0 to this field after initializing the message buffers and Control 1 (CTRL1) and Control 2 (CTRL2). FlexCAN performs no reception or transmission before this field becomes 0. Freeze mode cannot be entered when FlexCAN is in a low-power mode. 0b - No request 1b - Enter Freeze mode, if MCR[FRZ] = 1.
27 NOTRDY	FlexCAN Not Ready Indicates whether FlexCAN is in Disable mode, Stop mode or Freeze mode. When FlexCAN has exited these modes, this field becomes 0. Soft reset does not affect this field. 0b - FlexCAN is in Normal mode, Listen-Only mode, or Loopback mode. 1b - FlexCAN is in Disable mode, Stop mode, or Freeze mode.
26 WAKMSK	Wake-up Interrupt Mask Enables the wake-up interrupt generation under the Self Wake-Up mechanism. 0b - Disabled 1b - Enabled
25 SOFTRST	Soft Reset Resets internal state machines of FlexCAN and some memory-mapped registers. The CPU can write 1 to this field directly. This field also becomes 1 when global soft reset is requested at the chip level. Because soft reset is synchronous and must follow a request-and-acknowledge procedure across clock domains, it may take some time to propagate its effect fully. When reset is pending, this field remains 1; it automatically becomes 0 when reset completes. You can poll this field to know when the soft reset has completed. Soft reset cannot be applied when clocks are shut down in a low-power mode. Transfer the module out of the low-power mode before applying soft reset. Soft reset does not affect this field. **NOTE** This field becomes 0 within 2 CAN bits after assertion of this bit. 0b - No reset 1b - Soft reset affects reset registers
24 FRZACK	Freeze Mode Acknowledge Indicates whether FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore you can poll this field to know when FlexCAN has entered Freeze mode. If the Freeze mode request is negated, this field becomes 0 after the FlexCAN prescaler is running again. If Freeze mode is requested when FlexCAN is in a low-power mode, this field becomes 1 only when the low-power mode is exited. See Freeze mode. Soft reset does not affect this field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE FRZACK becomes 1 within 178 CAN bits following the Freeze mode request by the CPU. This field becomes 0 within 2 CAN bits after the Freeze mode request removal (see Protocol timing). 0b - Not in Freeze mode, prescaler running. 1b - In Freeze mode, prescaler stopped.
23 —	Reserved
22 SLFWAK	Self Wake-up Enables the Self Wake-up feature when FlexCAN is in a low-power mode other than Disable mode. When this feature is enabled, the FlexCAN module monitors the bus for a wake-up event (that is, a recessive-to-dominant transition). If a wake-up event is detected during Stop mode, FlexCAN generates, if enabled to do so, a wake-up interrupt to the CPU. The CPU can exit Stop mode globally and FlexCAN can request to resume the clocks. When FlexCAN is in a low-power mode other than Disable mode, this field cannot be written, as the module blocks it. When MCR[PNET_EN] = 1, this feature must be disabled. 0b - Disable 1b - Enable
21 WRNEN	Warning Interrupt Enable Enables the generation of the flags ESR1[TWRNINT] and ESR1[RWRNINT]. When this field is 1, TWRNINT and RWRNINT flags are set when the respective error counter transitions from less than 96 to greater than or equal to 96. When this field is 0, the TWRNINT and RWRNINT flags are always zero, independent of the values of the error counters. No warning interrupt is generated. This field can be written in Freeze mode only; the module blocks it in other modes. 0b - Disable 1b - Enable
20 LPMACK	Low-Power Mode Acknowledge Indicates whether FlexCAN is in a low-power mode (Disable mode, Stop mode). A low-power mode cannot be entered until all current transmission and reception processes have finished. The CPU can poll this field to know when FlexCAN has entered low-power mode. Soft reset does not affect this field. **NOTE** This field becomes 1 within 180 CAN bits after the low-power mode request by the CPU. This field becomes 0 within 2 CAN bits after the low-power mode request removal (see Protocol timing). When FlexCAN is in Pretended Networking mode, this field becomes 0 within 180 CAN bits after the low-power mode request removal.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not in a low-power mode 1b - In a low-power mode
19 WAKSRC	Wake-Up Source Determines whether the integrated low-pass filter is applied to the RX input that FlexCAN uses to detect recessive-to-dominant edges on the CAN bus. This filter can protect the RX CAN input from spurious wake-up. This field can be written only in Freeze mode. The module blocks it in other modes. 0b - No filter applied 1b - Filter applied
18 —	Reserved When writing to this field, always write the reset value.
17 SRXDIS	Self-Reception Disable Determines whether FlexCAN can receive frames transmitted by itself. If 1, frames transmitted by the module are not stored in any MB, regardless of whether the MB is programmed with an ID that matches the transmitted frame. No interrupt flag or interrupt signal is generated due to the frame reception. This field can be written only in Freeze mode; the module blocks it in other modes. 0b - Enable 1b - Disable
16 IRMQ	Individual RX Masking and Queue Enable Indicates whether RX matching process is based on individual masking and queue, or based on a masking scheme with RX Message Buffers Global Mask (RXMGMASK), Receive 14 Mask (RX14MASK), Receive 15 Mask (RX15MASK), and Legacy RX FIFO Global Mask (RXFGMASK). When this field is disabled, for backward compatibility with legacy applications, reading the Control and Status word locks the MB even if it is empty. This field can be written in Freeze mode only. The module blocks it in other modes. 0b - Disable 1b - Enable
15 DMA	DMA Enable Enables DMA. The DMA feature can only be used in Legacy RX FIFO or Enhanced RX FIFO, so MCR[RFEN] or ERFCR[ERFEN] must be 1. When DMA and RFEN are 1, IFLAG1[BUF5I] generates the DMA request, and no RX FIFO interrupt is generated. This field can be written in Freeze mode only; the module blocks it in other modes. 0b - Disable 1b - Enable
14 PNET_EN	Pretended Networking Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Enables Pretended Networking functionality. When in Stop mode, the PE subblock is kept operational. It can process RX message filtering as defined by the Pretended Networking configuration registers. See Receive process in Pretended Networking mode. This field can be written in Freeze mode only; the module blocks it in other modes. 0b - Disable 1b - Enable
13 LPRIOEN	Local Priority Enable Enables the local priority feature. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word. However, the actual transmitted ID is 11 bits for standard frames and 29 bits for extended frames. This field can be written only in Freeze mode; the module blocks it in other modes. This bit is provided for backward compatibility with legacy applications. 0b - Disable 1b - Enable
12 AEN	Abort Enable Enables the TX abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This field can be written only in Freeze mode; the module blocks it in other modes. **NOTE** When this field is 1, only use the abort mechanism (see Transmission abort mechanism) to update message buffers configured for transmission. **CAUTION** Writing the abort code into RX message buffers can cause unpredictable results when this field is 1. 0b - Disabled 1b - Enabled
11 FDEN	CAN FD Operation Enable Enables the CAN with flexible data rate (CAN FD) operation. This field can be written in Freeze mode only. FlexCAN can receive and transmit messages in CAN 2.0 format. If this field is enabled, FlexCAN can also receive and transmit messages in CAN FD format. FlexCAN can transmit FD frame format according to ISO 11898-1:2015. **NOTE** If the value of this field is 1, the Legacy RX FIFO Enable (MCR[RFEN]) field cannot be 1. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 —	Reserved
9-8 IDAM	ID Acceptance Mode Identifies the format of the Legacy RX FIFO ID filter table elements. This field configures all elements of the table at the same time; they are all the same format. See Legacy RX FIFO structure. This field can be written only in Freeze mode; the module blocks it in other modes. 00b - Format A: One full ID (standard and extended) per ID filter table element. 01b - Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID filter table element. 10b - Format C: Four partial 8-bit standard IDs per ID filter table element. 11b - Format D: All frames rejected.
7 —	Reserved
6-0 MAXMB	Number of the Last Message Buffer Defines the number of the last message buffer that takes part in the matching and arbitration processes. The reset value (0Fh) is equivalent to a 16-MB configuration. This field can be written only in Freeze mode; the module blocks it in other modes. **NOTE** You must write a value smaller than or equal to the number of available message buffers to this field, as described in FlexCAN memory partition for CAN FD. Additionally, the MAXMB value must consider the region of message buffers occupied by Legacy RX FIFO and its ID filters table space defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit, as described in Table 604.

68.6.2.3 Control 1 (CTRL1)

Offset

Register	Offset
CTRL1	4h

Function

Contains specific FlexCAN control features related to the CAN bus. These features include bit rate, programmable sampling point within an RX bit, Loopback mode, Listen-Only mode, Bus Off recovery behavior, and interrupt enabling (Bus-Off, Error, Warning). It also determines the division factor for the clock prescaler.

The CAN bit timing variables (CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW]) can also be configured in CAN Bit Timing (CBT), which extends the range of all these variables. If CBT[BTF] = 1, CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] become read-only.

If CTRL2[BTE] = 1, CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] are not used by the module. Instead, these fields are read as zero, and a write operation to them has no effect.

NOTE

When the CAN FD feature is enabled, do not use CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] for CAN bit timing. Instead use CBT[EPRESDIV], CBT[ERJW], CBT[EPSEG1], CBT[EPSEG2], and CBT[EPROPSEG].

The CAN bit variables in CTRL1 and in CBT are stored in the same internal register.

Soft reset does not affect the contents of this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BOFFMSK	ERRMSK	Reserved	LPB	TWRNMSK	RWRNMSK	0	0	SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 PRESDIV	Prescaler Division Factor Determines the ratio between the PE clock frequency and the serial clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The maximum value of this field is FFh, which gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes. Sclock frequency = PE clock frequency ÷ (PRESDIV + 1).
23-22 RJW	Resync Jump Width Defines the maximum number of time quanta that one resynchronization can change a bit time. One time quantum is equal to one Sclock period. The valid programmable values are 0–3. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes. Resync Jump Width = RJW + 1.
21-19 PSEG1	Phase Segment 1 Defines the length of phase segment 1 in the bit time. The valid programmable values are 0–7. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes. PhaseBuffer Segment 1 = (PSEG1 + 1) × Time Quanta.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-16 PSEG2	Phase Segment 2 Defines the length of phase segment 2 in the bit time. The valid programmable values are 1–7. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time Quanta.
15 BOFFMSK	Bus Off Interrupt Mask Provides a mask for the Bus Off interrupt ESR1[BOFFINT]. 0b - Interrupt disabled 1b - Interrupt enabled
14 ERRMSK	Error Interrupt Mask Provides a mask for the Error interrupt ESR1[ERRINT]. 0b - Interrupt disabled 1b - Interrupt enabled
13 —	Reserved **NOTE** Writeable only if the module is disabled. Otherwise the access type is read-only.
12 LPB	Loopback Mode Configures FlexCAN to operate in Loopback mode. In this mode, FlexCAN performs an internal loopback that can be used for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The RX CAN input pin is ignored and the TX CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field. It generates an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This field can be written only in Freeze mode; the module blocks it in other modes. **NOTE** In this mode, MCR[SRXDIS] cannot become 1, because it would impede the self-reception of a transmitted message. **NOTE** FDCTRL[TDCEN] and MCR[ETDCEN] must be 0 when this field is 1. 0b - Disabled 1b - Enabled
11	TX Warning Interrupt Mask

Table continues on the next page...

Table continued from the previous page...

Field	Function
TWRNMSK	Provides a mask for the TX Warning interrupt associated with the ESR1[TWRNINT] flag. When MCR[WRNEN] = 0, this field is read as 0. This field can be written only if MCR[WRNEN] = 1. 0b - Disabled 1b - Enabled
10 RWRNMSK	RX Warning Interrupt Mask Provides a mask for the RX Warning interrupt associated with the ESR1[RWRNINT] flag. When MCR[WRNEN] = 0, this field is read as 0. This field can be written only if MCR[WRNEN] = 1. 0b - Disabled 1b - Enabled
9 —	Reserved
8 —	Reserved
7 SMP	CAN Bit Sampling Determines the sampling mode of CAN bits at the RX input. This field can be written in Freeze mode only; the module blocks it in other modes. **NOTE** For proper operation, to write 1 to this field, you must guarantee a minimum value of two time quanta in CTRL1[PSEG1] (or CBT[EPSEG1]).This bit cannot become 1 when CAN FD is enabled (MCR[FDEN] = 1). 0b - One sample is used to determine the bit value. 1b - Three samples are used to determine the value of the received bit: the regular one (sample point) and two preceding samples. A majority rule is used.
6 BOFFREC	Bus Off Recovery Determines how FlexCAN recovers from Bus Off state. If 0, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If 1, automatic recovering from Bus Off is disabled. The module remains in Bus Off state until you write 1 to this field. If this field becomes 0 before 128 sequences of 11 recessive bits are detected on the CAN bus, Bus Off recovery happens as if this field had never become 1. If this field becomes 0 after 128 sequences of 11 recessive bits occurred, FlexCAN resynchronizes to the bus. It waits for 11 recessive bits before joining the bus. After this field becomes 0, it can become 1 again during Bus Off, but it will only be effective the next time the module enters Bus Off. If this field becomes 0 when the module is in Bus Off, writing 1 to this field is not effective for the current Bus Off recovery. **NOTE** See Bus Off in the CAN Protocol standard (ISO 11898-1:2015) for details.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enabled 1b - Disabled
5 TSYN	Timer Sync Enables a mechanism that resets the free-running timer each time a message is received in message buffer 0. This feature provides the means to synchronize multiple FlexCAN stations with a special "SYNC" message (that is, global network time). If MCR[RFEN] = 1 (Legacy RX FIFO enabled), the first available message buffer, according to CTRL2[RFFN], is used for timer synchronization instead of MB0. This field can be written in Freeze mode only; the module blocks it in other modes. 0b - Disable 1b - Enable
4 LBUF	Lowest Buffer Transmitted First Determines the ordering mechanism for message buffer transmission. When 1, MCR[LPRIOEN] does not affect the priority arbitration. This field can be written in Freeze mode only; the module blocks it in other modes. 0b - Buffer with highest priority is transmitted first. 1b - Lowest number buffer is transmitted first.
3 LOM	Listen-Only Mode Configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in Error Counter (ECR) are frozen, and the module operates in CAN Error Passive mode. Only messages acknowledged by another CAN station are received. If FlexCAN detects an unacknowledged message, it flags a BIT0 error without changing the receive error counter (ECR[RXERRCNT]), as if it is trying to acknowledge the message. FlexCAN acknowledges Listen-Only mode when ESR1[FLTCONF] = 1, indicating the Error Passive state. There can be some delay between the Listen-Only mode request and its acknowledgment. This field can be written in Freeze mode only; the module blocks it in other modes. 0b - Listen-Only mode is deactivated. 1b - FlexCAN module operates in Listen-Only mode.
2-0 PROPSEG	Propagation Segment Defines the length of the propagation segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode; the module blocks it in other modes. Propagation segment time = (PROPSEG + 1) × Time Quanta. One Time Quantum = one Sclock period.

68.6.2.4 Free-Running Timer (TIMER)

Offset

Register	Offset
TIMER	8h

Function

Represents a 16-bit free-running counter that the CPU can read and write. The timer starts from 0h after reset, counts linearly to FFFFh, and wraps around.

The CAN bit clock increments the timer, which defines the baud rate on the CAN bus. During a message transmission or reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable, Stop, Pretended Networking, and Freeze modes.

The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the timestamp entry in a message buffer after a successful reception or transmission of a message.

If CTRL1[TSYN] = 1, the timer is reset whenever a message is received in the first available message buffer, according to CTRL2[RFFN].

The CPU can write to this register anytime. However, if the write occurs simultaneously with the timer being reset by a reception in the first message buffer, the write value is discarded.

Reading this register affects the message buffer unlocking procedure (see Message buffer lock mechanism).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-0 TIMER	Timer Value Contains the free-running counter value.

68.6.2.5 RX Message Buffers Global Mask (RXMGMASK)

Offset

Register	Offset
RXMGMASK	10h

Function

Masks the filter bits of all RX message buffers, excluding MB14 and MB15, which have individual mask registers.

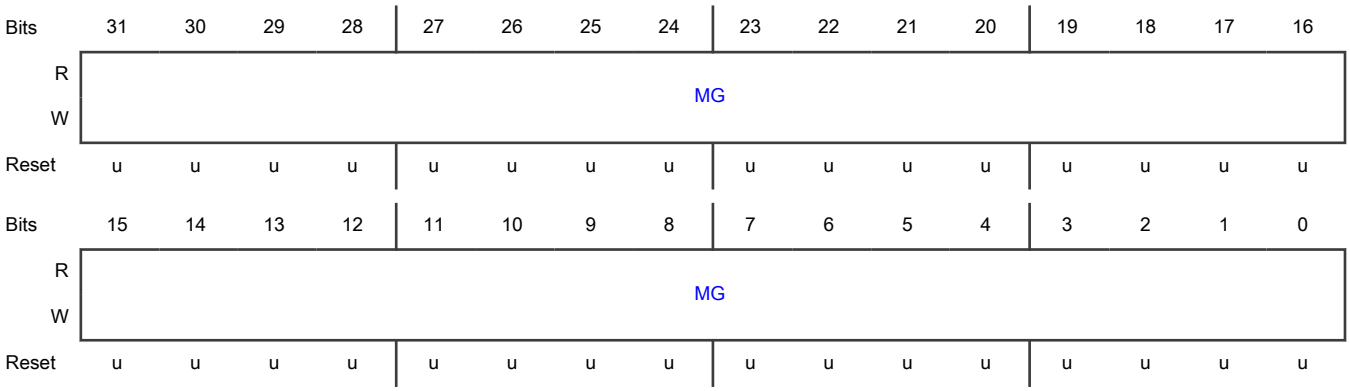
This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When MCR[IRMQ] is 0, RXMGMASK is always in effect. The bits in RXMGMASK[MG] mask the MB filter bits.
- When MCR[IRMQ] is 1, RXMGMASK has no effect. The bits in RXMGMASK[MG] do not mask the MB filter bits.

This register can only be written in Freeze mode; the module blocks it in other modes.

Diagram



Fields

Field	Function
31-0 MG	Global Mask for RX Message Buffers Masks the message buffer filter bits. The alignment with the ID word of the message buffer is imperfect. The two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the MB. The following table shows which MG bits mask each MB filter field.

SMB[RTR] [1]	CTRL2[RRS]	CTRL2[EACEN]	Message buffer filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
0	-	0	Note [2]	Note [3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th rowspan="2">SMB[RTR] [1]</th><th rowspan="2">CTRL2[RRS]</th><th rowspan="2">CTRL2[EACEN]</th><th colspan="4">Message buffer filter fields</th></tr>
<tr><th>MB[RTR]</th><th>MB[IDE]</th><th>MB[ID]</th><th>Reserved</th></tr>
<tr><td>1</td><td>1</td><td>1</td><td>MG[31]</td><td>MG[30]</td><td>MG[28:0]</td><td>MG[29]</td></tr>
</table>
1. RTR bit of the incoming frame. It is saved into an auxiliary MB called RX serial message buffer (RX SMB).
2. If CTRL2[EACEN] is 0, the RTR bit of MB is never compared with the RTR bit of the incoming frame.
3. If CTRL2[EACEN] is 0, the IDE bit of MB is always compared with the IDE bit of the incoming frame.

0b - The corresponding bit in the filter is "don't care."

1b - The corresponding bit in the filter is checked.
</td></tr>
</table>

68.6.2.6 Receive 14 Mask (RX14MASK)

Offset

Register	Offset
RX14MASK	14h

Function

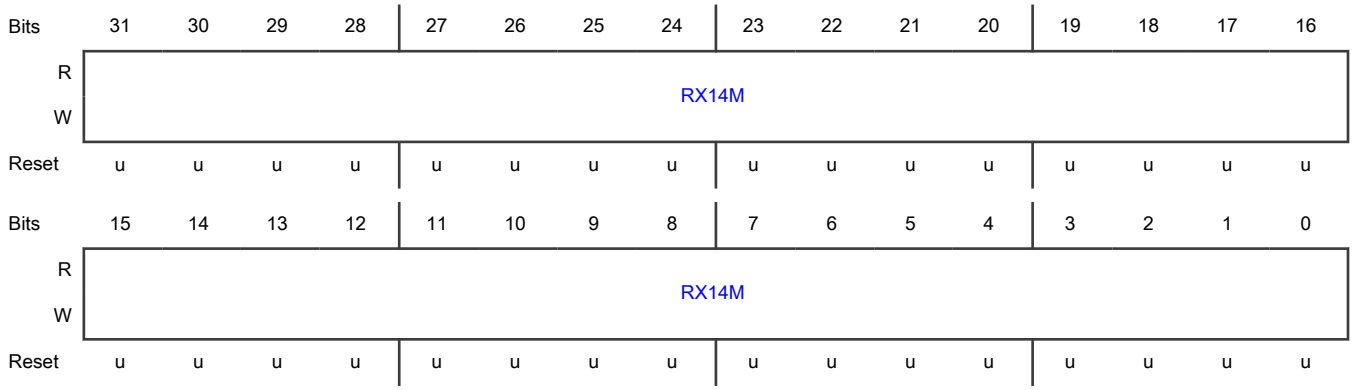
Masks the filter fields of MB14.

This register is located in RAM.

RX14MASK is provided for legacy application support. When MCR[IRMQ] = 1, RX14MASK has no effect.

This register can only be programmed when the module is in Freeze mode; the module blocks it in other modes.

Diagram



Fields

Field	Function
31-0 RX14M	RX Buffer 14 Mask Bits Masks the corresponding MB14 filter field in the same way that RX Message Buffers Global Mask (RXMGMASK) masks the filters of the other message buffers. 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

68.6.2.7 Receive 15 Mask (RX15MASK)

Offset

Register	Offset
RX15MASK	18h

Function

Masks the filter fields of MB15.

This register is located in RAM.

RX15MASK is provided for legacy application support. When MCR[IRMQ] = 1, RX15MASK has no effect.

This register can be programmed only when the module is in Freeze mode; the module blocks it in other modes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RX15M															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX15M															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 RX15M	RX Buffer 15 Mask Bits Masks the corresponding MB15 filter field in the same way that RX Message Buffers Global Mask (RXMGMASK) masks the filters of other message buffers.

Table continues on the next page...

Field	Function
	0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

68.6.2.8 Error Counter (ECR)

Offset

Register	Offset
ECR	1Ch

Function

Contains error counters for received and transmitted messages.

TXERRCNT and RXERRCNT consider all errors in both CAN FD and non-FD message formats. TXERRCNT_FAST and RXERRCNT_FAST count only the errors that occur in the data phase of CAN FD frames that have BRS = 1.

The Fault Confinement state (ESR1[FLTCONF]) is updated based on TXERRCNT and RXERRCNT counters only. The rules for increasing and decreasing these counters are described in the CAN protocol and are entirely implemented in FlexCAN.

The basic rules for FlexCAN bus state transitions are:

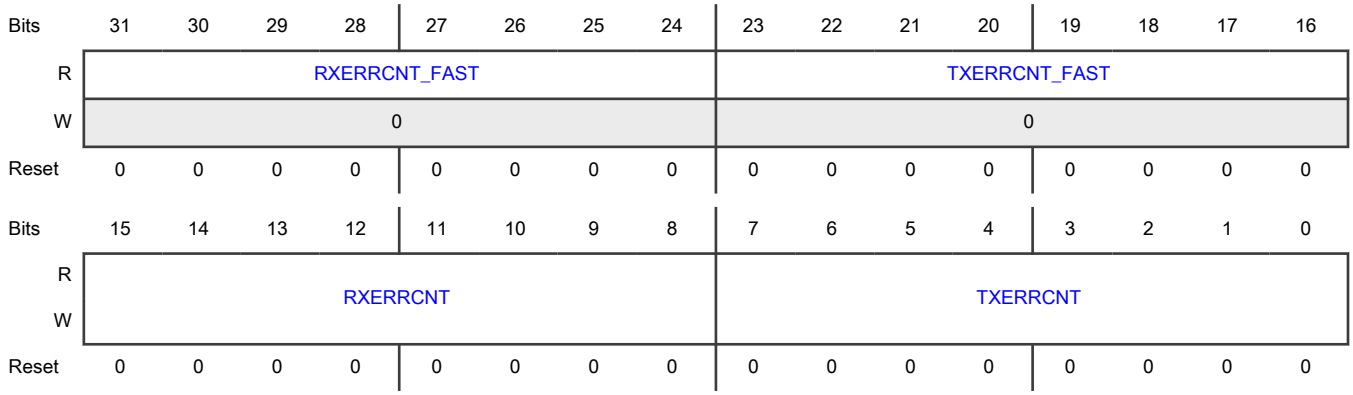
- If the value of TXERRCNT or RXERRCNT becomes greater than or equal to 128, ESR1[FLTCONF] is updated to reflect Error Passive state.
- If the state of FlexCAN is Error Passive, and TXERRCNT or RXERRCNT decrements to a value less than 128 when the other already satisfies this condition, ESR1[FLTCONF] is updated to reflect Error Active state.
- If the value of TXERRCNT becomes greater than 255, ESR1[FLTCONF] is updated to reflect Bus Off state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in Bus Off, TXERRCNT is cascaded with another internal counter to count the occurrences of 11 consecutive recessive bits on the bus. TXERRCNT is reset to zero. It counts in a manner where the internal counter counts 11 such bits and then wraps around when incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, ESR1[FLTCONF] is updated to Error Active, and both error counters are reset to zero. Upon any instance of a dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNT_FAST counter is frozen during Bus Off.
- If only one node is operating during system startup, its TXERRCNT increases upon each attempted message transmission, as a result of acknowledge errors (indicated by ESR1[ACKERR]). After the transition to Error Passive state, TXERRCNT no longer increments upon acknowledge errors. The chip never goes into the Bus Off state.
- If RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected when being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to return to the Error Active state.
- TXERRCNT_FAST and RXERRCNT_FAST error counter values increment and decrement based on errors detected only in the data phase of CAN FD frames that have BRS = 1. These counters follow the same increment and decrement rules as TXERRCNT and RXERRCNT. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen when FlexCAN is in the Bus Off state. They are reset when FlexCAN leaves the Bus Off state and resume counting after FlexCAN returns to the Error Active state.
- When FlexCAN is in Pretended Networking mode, RXERRCNT and RXERRCNT_FAST keep counting errors, and error flags are stored. TXERRCNT and TXERRCNT_FAST preserve their values and do not change, because no transmission occurs in Pretended Networking mode. Error counters and error flags that changed values in Pretended Networking mode

are updated in this register and in Error and Status 1 (ESR1) when FlexCAN returns to Normal mode. If FlexCAN is in Pretended Networking mode, the FAST error flags in ESR1 are not set.

NOTE

See Fault confinement in the CAN Protocol standard (ISO 11898-1:2015) for details.

Diagram



Fields

Field	Function
31-24 RXERRCNT_FAST	Receive Error Counter for Fast Bits Counts errors detected in the data phase of received CAN FD messages that have BRS = 1. This field is read-only except in Freeze mode, when the CPU can write an 8-bit zero value only.
23-16 TXERRCNT_FAST	Transmit Error Counter for Fast Bits Counts errors detected in the data phase of transmitted CAN FD messages that have BRS = 1. This field is read-only except in Freeze mode, when the CPU can write an 8-bit zero value only.
15-8 RXERRCNT	Receive Error Counter Counts all errors detected in received messages. This field is read-only except in Freeze mode, when the CPU can write to it.
7-0 TXERRCNT	Transmit Error Counter Counts all errors detected in transmitted messages. This field is read-only except in Freeze mode, when the CPU can write to it.

68.6.2.9 Error and Status 1 (ESR1)

Offset

Register	Offset
ESR1	20h

Function

Reports various error conditions detected in the reception and transmission of a CAN frame. This register provides status information about the chip, and is the source of some interrupts to the CPU. The reported error conditions are:

NOTE

Reading host can clear these fields.

- Errors detected in CAN frames of any format:
  - BIT1ERR
  - BIT0ERR
  - ACKERR
  - CRCERR
  - FRMERR
  - STFERR
- Errors detected in the data phase of CAN FD frames with the BRS bit set only:
  - BIT1ERR_FAST
  - BIT0ERR_FAST
  - CRCERR_FAST
  - FRMERR_FAST
  - STFERR_FAST

One or more error flags may report an error detected in a single CAN frame. To account for more error events occurring in subsequent frames when the CPU does not attempt to read this register, error reporting is cumulative.

Status flags:

- TXWRN
- RXWRN
- IDLE
- TX
- FLTCONF
- RX
- SYNCH

Interrupt flags:

- BOFFINT
- BOFFDONEINT
- ERRINT
- ERRINT_FAST
- WAKINT
- TWRNINT
- RWRNINT

The CPU should follow this procedure when servicing interrupt requests generated by these flags:

1. Read this register to capture all error condition and status flags. This action clears the respective flags that were set since the last read access.

2. Write 1 to clear the interrupt flag that triggered the interrupt request.
3. Write 1 to clear the ERROVR flag, if it is set.

Starting from all error flags cleared, a first error event sets either ERRINT or ERRINT_FAST (provided the corresponding mask bit is 1). If other error events in subsequent frames occur before the CPU serves the interrupt request, the ERROVR flag is set to indicate that errors from different frames have accumulated.

Table 610. CAN bus status

SYNCH	IDLE	TX	RX	FlexCAN state
0	0	0	0	Not synchronized to CAN bus
1	1	X	X	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

NOTE

See Fault confinement in the CAN Protocol standard (ISO 11898-1:2015) for details.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BIT1ERR...	BIT0ERR...	0	CRCERR_...	FRMERR_...	STFERR_...	0		0	0	ERROVR	ERRINT_...	BOFFDON...	SYNCH	TWRNINT	RWRNINT
W											W1C	W1C	W1C		W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	WAKINT
W														W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 BIT1ERR_FAST	Fast Bit1 Error Flag Indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames that have BRS = 1. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit transmitted as recessive is received as dominant.
30	Fast Bit0 Error Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
BIT0ERR_FAST	Indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames that have BRS = 1. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit transmitted as dominant is received as recessive.
29 —	Reserved
28 CRCERR_FAST	Fast Cyclic Redundancy Check Error Flag Indicates that the receiver node has detected a CRC error in the CRC field of CAN FD frames that have BRS = 1. This error means that the calculated CRC is different from the received CRC. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A CRC error occurred since last read of this register.
27 FRMERR_FAST	Fast Form Error Flag Indicates whether the receiver node has detected a form error in the data phase of CAN FD frames that have BRS = 1. This error means that a fixed-form bit field contains at least one illegal bit. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A form error occurred since last read of this register.
26 STFERR_FAST	Fast Stuffing Error Flag Indicates that a stuffing error has been detected in the data phase of CAN FD frames that have BRS = 1. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A stuffing error occurred since last read of this register.
25-24 —	Reserved
23 —	Reserved
22 —	Reserved
21 ERROVR	Error Overrun Flag Indicates that an error condition occurred when any error flag is already set.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No overrun 1b - Overrun
20 ERRINT_FAST	Fast Error Interrupt Flag Indicates that at least one error flag detected in the data phase of CAN FD frames that have BRS = 1 (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST, or STFERR_FAST) is set. If CTRL2[ERRMSK_FAST] = 1, an interrupt is generated to the CPU. 0b - No such occurrence. 1b - Error flag set in the data phase of CAN FD frames that have BRS = 1.
19 BOFFDONEINT	Bus Off Done Interrupt Flag Indicates whether ECR[TXERRCNT] has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If CTRL2[BOFFDONEMSK] = 1, an interrupt is generated to the CPU. 0b - No such occurrence 1b - FlexCAN module has completed Bus Off process.
18 SYNCH	CAN Synchronization Status Flag Indicates whether FlexCAN is synchronized to the CAN bus and able to participate in the communication process. FlexCAN sets and clears this flag. See the table in Error and Status 1 (ESR1). 0b - Not synchronized 1b - Synchronized
17 TWRNINT	TX Warning Interrupt Flag Indicates whether TX error counter changed from less than 96 to greater than or equal to 96. If MCR[WRNEN] = 1, this flag is set when the TXWRN flag transitions from 0 to 1, meaning that the TX error counters reached 96. If CTRL1[TWRNMSK] = 1, an interrupt is sent to the CPU. When MCR[WRNEN] = 0, this flag is masked. The CPU must clear this flag before writing 0 to MCR[WRNEN]. Otherwise, this flag is set when MCR[WRNEN] = 1 again. Writing 0 has no effect. This flag is not generated when in the Bus Off state. This flag is not updated during Freeze mode. When FlexCAN returns to Normal mode from Pretended Network mode (see Receive process in Pretended Networking mode), this bit is not updated. 0b - No such occurrence 1b - TX error counter changed from less than 96 to greater than or equal to 96.
16 RWRNINT	RX Warning Interrupt Flag Indicates whether the RX error counter changed from less than 96 to greater than or equal to 96. If MCR[WRNEN] = 1, this flag is set when the RXWRN flag transitions from 0 to 1, meaning that the RX error counters reached 96. If CTRL1[RWRNMSK] = 1, an interrupt is sent to the CPU. When MCR[WRNEN] = 0, this flag is masked. The CPU must clear this flag before writing 0 to MCR[WRNEN]. Otherwise, this flag is set when MCR[WRNEN] = 1 again. Writing 0 has no effect.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This flag is not updated during Freeze mode. When FlexCAN returns to Normal mode from Pretended Network mode (see Receive process in Pretended Networking mode), this bit is updated to reflect the RX error counter state. 0b - No such occurrence 1b - RX error counter changed from less than 96 to greater than or equal to 96.
15 BIT1ERR	Bit1 Error Flag Indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. This flag is updated when FlexCAN returns to Normal mode from Pretended Network mode. **NOTE** A transmitter does not set this flag for an arbitration field or ACK slot. It is not set for a node sending a error passive flag that detects dominant bits. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit sent as recessive is received as dominant.
14 BIT0ERR	Bit0 Error Flag Indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. This field is updated when FlexCAN returns to Normal mode from Pretended Network mode. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit sent as dominant is received as recessive.
13 ACKERR	Acknowledge Error Flag Indicates whether the transmitter node has detected an acknowledge error. This error means that a dominant bit has not been detected during the ACK SLOT. This flag is updated when FlexCAN returns to Normal mode from Pretended Network mode. After a read operation, the field's value clears to 0. 0b - No error 1b - Error occurred since last read of this register.
12 CRCERR	Cyclic Redundancy Check Error Flag Indicates whether the receiver node has detected a cyclic redundancy check (CRC) error either in a non-FD message or in the arbitration or data phase of a frame in CAN FD format. This error means that the calculated CRC is different from the received. This flag is updated when FlexCAN returns to Normal mode from Pretended Network mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	After a read operation, the field's value clears to 0. 0b - No error 1b - Error occurred since last read of this register.
11 FRMERR	Form Error Flag Indicates whether a form error has been detected in a non-FD message or in the arbitration or data phase of an FD message by the receiver node. This error means that a fixed-form field contains at least one illegal bit. This flag is updated when FlexCAN returns to Normal mode from Pretended Network mode. After a read operation, the field's value clears to 0. 0b - No error 1b - Error occurred since last read of this register.
10 STFERR	Stuffing Error Flag Indicates that a stuffing error has been detected in a non-FD message or in the arbitration or data phase of an FD message by the receiver node. This flag is updated when FlexCAN returns to Normal mode from Pretended Network mode. After a read operation, the field's value clears to 0. 0b - No error 1b - Error occurred since last read of this register.
9 TXWRN	TX Error Warning Flag Indicates when repetitive errors occur during message transmission. Only the value of ECR[TXERRCNT] affects this flag. This flag is not updated during Freeze mode. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - TXERRCNT is 96 or greater.
8 RXWRN	RX Error Warning Flag Indicates when repetitive errors occur during message reception. Only the value of ECR[RXERRCNT] affects this flag. This flag is not updated during Freeze mode. Additionally, this flag is updated when FlexCAN returns to Normal mode from Pretended Networking mode. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - RXERRCNT is greater than or equal to 96.
7 IDLE	Idle Indicates whether CAN bus is in IDLE state. See Table 610. 0b - Not IDLE

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - IDLE
6 TX	FlexCAN In Transmission Indicates whether FlexCAN is transmitting a message. See Table 610. 0b - Not transmitting 1b - Transmitting
5-4 FLTCONF	Fault Confinement State Indicates the confinement state of FlexCAN. If CTRL1[LOM] = 1, after a delay that depends on the CAN bit timing, this field indicates Error Passive. The same delay affects the way that this field reflects an update to ECR register by the CPU. It may be necessary to wait up to one CAN bit time for coherence to be restored. Soft reset affects this field, but if CTRL1[LOM] = 1, its reset value lasts for only one CAN bit. After that time, this field reports Error Passive. 00b - Error Active 01b - Error Passive 1xb - Bus Off
3 RX	FlexCAN in Reception Flag Indicates whether FlexCAN is receiving a message. See the table in Error and Status 1 (ESR1). 0b - Not receiving 1b - Receiving
2 BOFFINT	Bus Off Interrupt Flag Indicates whether FlexCAN has entered Bus Off state. If CTRL1[BOFFMSK]) = 1, an interrupt is generated to the CPU. Writing 0 to this field has no effect. 0b - No such occurrence. 1b - FlexCAN module entered Bus Off state.
1 ERRINT	Error Interrupt Flag Indicates that at least one of the error flags (ESR1[BIT1ERR], ESR1[BIT0ERR], ESR1[ACKERR], ESR1[CRCERR], ESR1[FRMERR], or ESR1[STFERR]) is set. If the corresponding mask CTRL1[ERRMSK] = 1, an interrupt is generated to the CPU. Writing 0 to this field has no effect. 0b - No such occurrence. 1b - Indicates setting of any error flag in the Error and Status register.
0 WAKINT	Wake-up Interrupt Flag Generates an interrupt to the CPU when a recessive-to-dominant transition is detected on the CAN bus and MCR[WAKMSK] = 1. This field applies when FlexCAN is in low-power mode during Self Wake-up:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• Stop mode When MCR[SLFWAK] = 0, this flag is masked. The CPU must clear this flag before disabling the field. Otherwise, it is set when MCR[SLFWAK] becomes 1 again. Writing 0 has no effect. 0b - No such occurrence. 1b - Indicates that a recessive-to-dominant transition was received on the CAN bus.

68.6.2.10 Interrupt Masks 1 (IMASK1)

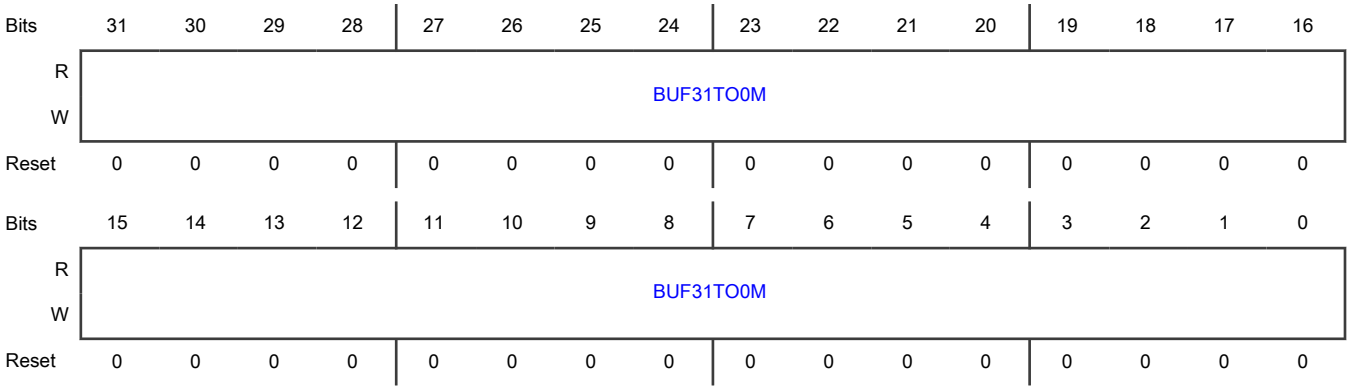
Offset

Register	Offset
IMASK1	28h

Function

Masks interrupt flags. This register allows any of the 32 message buffer interrupts to be enabled or disabled for MB31–MB0. It contains one interrupt mask bit per buffer. This configuration allows the CPU to determine which buffer generates an interrupt after a successful transmission or reception when the corresponding Interrupt Flags 1 (IFLAG1) flag is set.

Diagram



Fields

Field	Function
31-0 BUF31TO0M	Buffer MBi Mask Enables or disables the corresponding FlexCAN message buffer interrupt for MB31–MB0. **NOTE** If the corresponding Interrupt Flags 1 (IFLAG1) flag is set, writing 1 or 0 to a field in IMASK1 can set or clear an interrupt request.

Table continues on the next page...

Field	Function
	0b - The corresponding buffer interrupt is disabled. 1b - The corresponding buffer interrupt is enabled.

68.6.2.11 Interrupt Flags 1 (IFLAG1)

Offset

Register	Offset
IFLAG1	30h

Function

Contains the flags for the 32 message buffer interrupts for MB31–MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding Interrupt Masks 1 (IMASK1) bit is set, an interrupt is generated. There is an exception when DMA for Legacy RX FIFO is enabled, as described below.

The BUF7I–BUF5I flags also represent Legacy FIFO interrupts when the Legacy RX FIFO is enabled. When MCR[RFEN] is 1 and MCR[DMA] is 0, the function of the eight least significant interrupt flags changes:

- BUF7I, BUF6I, and BUF5I indicate operating conditions of the Legacy FIFO.
- BUF4I–BUF1I fields are reserved.
- BUF0I empties the Legacy FIFO.

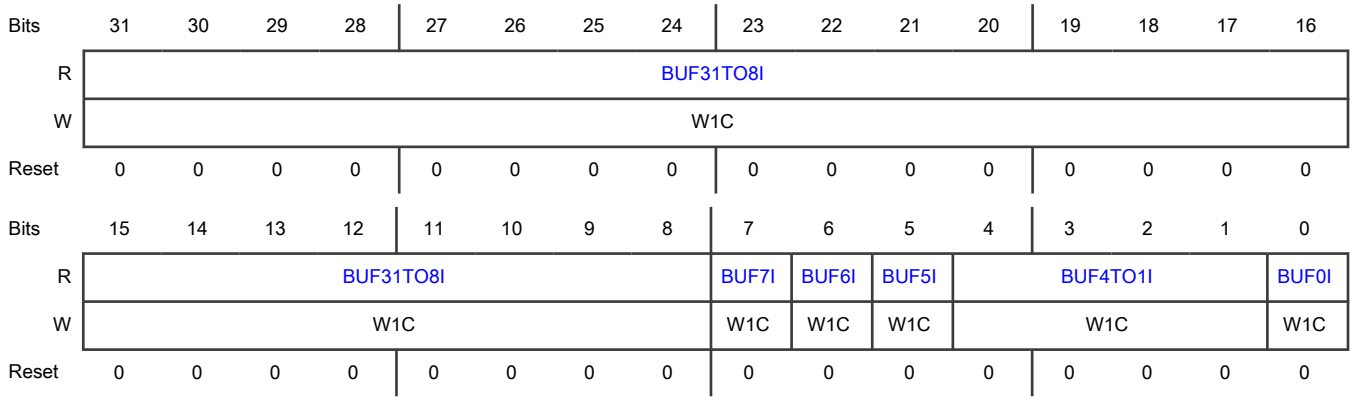
Before writing 1 to MCR[RFEN], the CPU must service the IFLAG flags set in the Legacy RX FIFO region; see Legacy RX FIFO. Otherwise, these IFLAG flags mistakenly show the related message buffers now belonging to Legacy FIFO as having contents to be serviced. When MCR[RFEN] is 0, the Legacy FIFO flags must be cleared. The same care must be taken when a CTRL2[RFFN] value is selected, extending Legacy RX FIFO filters beyond MB7. For example, when RFFN is 8h, Legacy RX FIFO filters occupy the MB23–MB0 range, and related IFLAG flags must be cleared.

When MCR[RFEN] and MCR[DMA] are 1 (DMA feature for Legacy RX FIFO enabled), the function of the eight least significant interrupt flags (BUF7I–BUF0I) changes to support DMA operation. BUF7I, BUF6I, and BUF4I–BUF1I are not used.

BUF5I indicates the operating condition of the Legacy FIFO, and BUF0I empties the Legacy FIFO. Moreover, BUF5I does not generate a CPU interrupt, but it does generate a DMA request. IMASK1 bits in the Legacy RX FIFO region are not considered when bit MCR[DMA] = 1. In addition, the CPU must not clear the BUF5I flag when DMA is enabled. Before writing 1 to MCR[DMA], the CPU must service the IFLAG flags set in the Legacy RX FIFO region. When MCR[DMA] is 0, the Legacy FIFO must be empty. Legacy FIFO must be disabled when MCR[FDEN] = 1.

Before updating MCR[MAXMB], the CPU must service the IFLAG1 flags whose MB value is greater than the MCR[MAXMB] to be updated. Otherwise, those flags remain set and are inconsistent with the number of message buffers available.

Diagram



Fields

Field	Function
31-8 BUF31TO8I	Buffer MBi Interrupt Flags the corresponding FlexCAN message buffer interrupt for MB31–MB8. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.
7 BUF7I	Buffer MB7 Interrupt or Legacy RX FIFO Overflow Flags the interrupt for MB7 when MCR[RFEN] = 0 (Legacy RX FIFO disabled). **NOTE** When a CPU write changes the value of MCR[RFEN], FlexCAN clears this flag. When MCR[RFEN] = 1, this flag represents a Legacy RX FIFO overflow. In this case, the flag indicates that a message was lost because the Legacy RX FIFO is full. When the Legacy RX FIFO is full and a message buffer captures the message, this flag is not set. 0b - No occurrence of MB7 completing transmission or reception, or no FIFO overflow. 1b - MB7 completed transmission or reception, or FIFO overflow.
6 BUF6I	Buffer MB6 Interrupt or Legacy RX FIFO Warning Flags the interrupt for MB6 when MCR[RFEN] = 0 (Legacy RX FIFO disabled). **NOTE** When a CPU write changes the value of MCR[RFEN], FlexCAN clears this flag. When MCR[RFEN] = 1, this flag represents a Legacy RX FIFO warning. In this case, the flag indicates when the number of unread messages within the Legacy RX FIFO is increased to five from four due to the reception of a new message. In other words, the Legacy RX FIFO is almost full. If this flag is cleared when there are more than four unread messages, it does not set again until the number of unread messages in the Legacy RX FIFO decreases to four or fewer. 0b - No occurrence of MB6 completing transmission or reception, or FIFO not almost full.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - MB6 completed transmission or reception, or FIFO almost full.
5 BUF5I	Buffer MB5 Interrupt or Frames available in Legacy RX FIFO Flags the interrupt for MB5 when MCR[RFEN] = 0 (Legacy RX FIFO disabled). **NOTE** When a CPU write changes the value of MCR[RFEN], FlexCAN clears this flag. When MCR[RFEN] = 1, the BUF5I flag represents frames available in Legacy RX FIFO. In this case, the flag indicates that at least one frame is available to be read from the Legacy RX FIFO. When MCR[DMA] = 1, this flag generates a DMA request. The CPU must not clear this field by writing 1 to BUF5I. 0b - No occurrence of completed transmission or reception, or no frames available 1b - MB5 completed transmission or reception, or frames available
4-1 BUF4TO1I	Buffer MBi Interrupt or Reserved Flags the interrupts for MB4–MB1 when MCR[RFEN] = 0 (Legacy RX FIFO disabled). **NOTE** When a CPU write changes the value of MCR[RFEN], FlexCAN clears these flags. When MCR[RFEN] = 1, the BUF4TO1I flags are reserved. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.
0 BUF0I	Buffer MB0 Interrupt or Clear Legacy FIFO bit Flags the interrupt for MB0 when MCR[RFEN] = 0 (Legacy RX FIFO disabled). If MCR[RFEN] = 1, this field is used to trigger the clear Legacy FIFO operation. This operation empties the Legacy FIFO contents. Before performing this operation, the CPU must service all Legacy FIFO-related IFLAG flags. When MCR[DMA] = 1, this operation also clears the BUF5I flag, aborting the DMA request. The clear Legacy FIFO operation occurs when the CPU writes 1 to BUF0I. This operation is only allowed in Freeze mode; the module blocks it in other conditions. 0b - MB0 has no occurrence of successfully completed transmission or reception. 1b - MB0 has successfully completed transmission or reception.

68.6.2.12 Control 2 (CTRL2)

Offset

Register	Offset
CTRL2	34h

Function

Complements Control 1 (CTRL1), providing control bits for memory write-access in Freeze mode. This register extends Legacy FIFO filter quantity, and adjusts the operation of internal FlexCAN processes such as matching and arbitration.

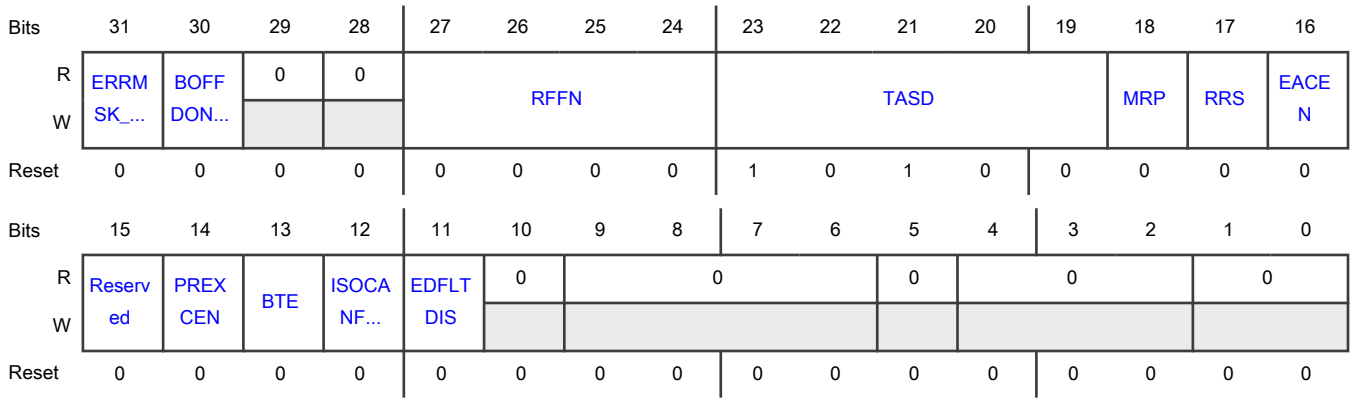
Soft reset does not affect the contents of this register.

Table 611 shows how the value of CTRL2[RFFN] determines the Legacy RX FIFO filter structure.

Table 611. Possible Legacy RX FIFO filter structures

RFFN[3:0]	Number of Legacy RX FIFO filter elements	Message buffers occupied by Legacy RX FIFO and ID filter table	Remaining available message buffers	Legacy RX FIFO ID filter table elements affected by RX individual masks	Legacy RX FIFO ID filter table elements affected by Legacy RX FIFO global mask
0h	8	MB 0–7	MB 8–31	Elements 0–7	None
1h	16	MB 0–9	MB 10–31	Elements 0–9	Elements 10–15
2h	24	MB 0–11	MB 12–31	Elements 0–11	Elements 12–23
3h	32	MB 0–13	MB 14–31	Elements 0–13	Elements 14–31
4h	40	MB 0–15	MB 16–31	Elements 0–15	Elements 16–39
5h	48	MB 0–17	MB 18–31	Elements 0–17	Elements 18–47
6h	56	MB 0–19	MB 20–31	Elements 0–19	Elements 20–55
7h	64	MB 0–21	MB 22–31	Elements 0–21	Elements 22–63
8h	72	MB 0–23	MB 24–31	Elements 0–23	Elements 24–71
9h	80	MB 0–25	MB 26–31	Elements 0–25	Elements 26–79
Ah	88	MB 0–27	MB 28–31	Elements 0–27	Elements 28–87
Bh	96	MB 0–29	MB 30–31	Elements 0–29	Elements 30–95
Ch	104	MB 0–31	none	Elements 0–31	Elements 32–103

Diagram



Fields

Field	Function
31 ERRMSK_FAST	Error Interrupt Mask for Errors Detected in the Data Phase of Fast CAN FD Frames Enables the ESR1[ERRINT_FAST] interrupt. 0b - Disable 1b - Enable
30 BOFFDONEMSK	Bus Off Done Interrupt Mask Enables the Bus Off Done interrupt, ESR1[BOFFDONEINT]. 0b - Disable 1b - Enable
29 —	Reserved
28 —	Reserved
27-24 RFFN	Number of Legacy Receive FIFO Filters Defines the number of Receive Legacy FIFO filters, as shown in Table 611. The chip determines the maximum selectable number of filters. Do not program this field with values that cause the number of message buffers occupied by Legacy RX FIFO and Legacy RX FIFO ID Filter to exceed MCR[MAXMB]. MCR[MAXMB] defines the number of message buffers present. This field can only be written in Freeze mode; the module blocks it in other modes. Each group of eight filters occupies a memory space equivalent to two message buffers. The more filters are implemented, the fewer message buffers are available. The Legacy RX FIFO occupies the memory space originally reserved for MB5–MB0. This field should be programmed with a value corresponding to a number of filters less than the number of available memory words. The number of available words can be calculated as follows: (SETUP_MB - 6) × 4 Where SETUP_MB is the smaller of the parameter NUMBER_OF_MB and MCR[MAXMB]. The number of remaining message buffers available is: (SETUP_MB - 8) - (RFFN × 2) If the number of Legacy RX FIFO filters programmed through RFFN exceeds the SETUP_MB value (memory space available), the exceeding ones are not functional. **NOTE** • The number of the last remaining available message buffers is the smaller of (NUMBER_OF_MB - 1) and MCR[MAXMB]. • If RX Individual Mask registers are not enabled, the Legacy RX FIFO Global Mask affects all Legacy RX FIFO filters.
23-19	Transmission Arbitration Start Delay

Table continues on the next page...

Table continued from the previous page...

Field	Function
TASD	Indicates by how many CAN bits the transmission arbitration process start point can be delayed from the first bit of CRC field on CAN bus. See TX arbitration start delay for details. This field can be written only in Freeze mode; the module blocks it in other modes.
18 MRP	Message Buffers Reception Priority Sets the priority for the matching process. This field can be written only in Freeze mode; the module blocks it in other modes. 0b - Matching starts from Legacy RX FIFO or Enhanced RX FIFO and continues on message buffers. 1b - Matching starts from message buffers and continues on Legacy RX FIFO or Enhanced RX FIFO.
17 RRS	Remote Request Storing Determines what the module does with a remote request. The remote request frame is submitted to a matching process. If this field is 1, the frame is stored in the corresponding message buffer in the same fashion as a data frame. No automatic remote response frame is generated. If this field is 0, an automatic remote response frame is generated if a message buffer with CODE = 1010b is found with the same ID. You can only write to this field in Freeze mode. The module blocks it in other modes. 0b - Generated 1b - Stored
16 EACEN	Entire Frame Arbitration Field Comparison Enable for RX Message Buffers Controls the comparison of IDE and RTR fields within RX message buffer filters with their corresponding bits in the incoming frame by the matching process. If enabled, the IDE and RTR fields of the RX message buffer are compared to their corresponding bits within the incoming frame (mask bits apply). If disabled, the IDE field of the RX message buffer filter is always compared and RTR is never compared despite mask bits. This field does not affect matching for Legacy RX FIFO or Enhanced RX FIFO. You can only write to this field in Freeze mode; the module blocks it in other modes. 0b - Disable 1b - Enable
15 —	Reserved When writing to this field, always write the reset value.
14 PREXCEN	Protocol Exception Enable Enables the protocol exception feature. You can only write to this field in Freeze mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE See Protocol exception event in the CAN Protocol standard (ISO 11898-1:2015) for details. 0b - Disabled 1b - Enabled
13 BTE	Bit Timing Expansion Enable Enables the use of Enhanced CAN Bit Timing Prescalers (EPRS), Enhanced Data Phase CAN Bit Timing (EDCBT), and Enhanced Nominal CAN Bit Timing (ENCBT) to configure the CAN bit timing segments, instead of using the bit timing fields of CAN Bit Timing (CBT), CAN FD Bit Timing (FDCBT), and Control 1 (CTRL1). If this field is 1: • CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] are read as zero. A write operation to these fields has no effect. • CBT[EPRESDIV], CBT[ERJW], CBT[EPROPSEG], CBT[EPSEG1], and CBT[EPSEG2], and the corresponding fields in CAN FD Bit Timing (FDCBT), are read as zero. A write operation to these fields has no effect. • ETDC[ETDCOFF], ETDC[ETDCEN], ETDC[ETDCFAIL], and ETDC[ETDCVAL] are used by FlexCAN instead of FDCTRL[TDCOFF], FDCTRL[TDCEN], FDCTRL[TDCFAIL], and FDCTRL[TDCVAL]. These fields are read as zero, and a write operation to them has no effect. • ETDC[TDMDIS] can be used to disable transceiver delay measurement. 0b - Disable 1b - Enable
12 ISOCANFDEN	ISO CAN FD Enable Enables the CAN FD protocol according to ISO specification (ISO 11898-1:2015) (see CAN FD ISO compliance). When disabled, FlexCAN operates using the non-ISO CAN FD protocol. You can only write to this field in Freeze mode. **NOTE** FlexCAN is able to transmit FD frame format according to CAN Protocol standard (ISO 11898-1:2015). 0b - Disable 1b - Enable
11 EDFLTDIS	Edge Filter Disable Disables the edge filter used during the Bus Integration state. When the Edge Filter is enabled, two consecutive nominal time quanta with dominant bus states are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of 11 consecutive recessive bits is restarted. The edge filter prevents dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD frame) from being mistaken for an idle condition.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	You can only write to this field in Freeze mode. **NOTE** See Bus Integration state in the CAN Protocol standard (ISO 11898-1:2015) for details. 0b - Enabled 1b - Disabled
10 —	Reserved
9-6 —	Reserved
5 —	Reserved
4-2 —	Reserved
1-0 —	Reserved

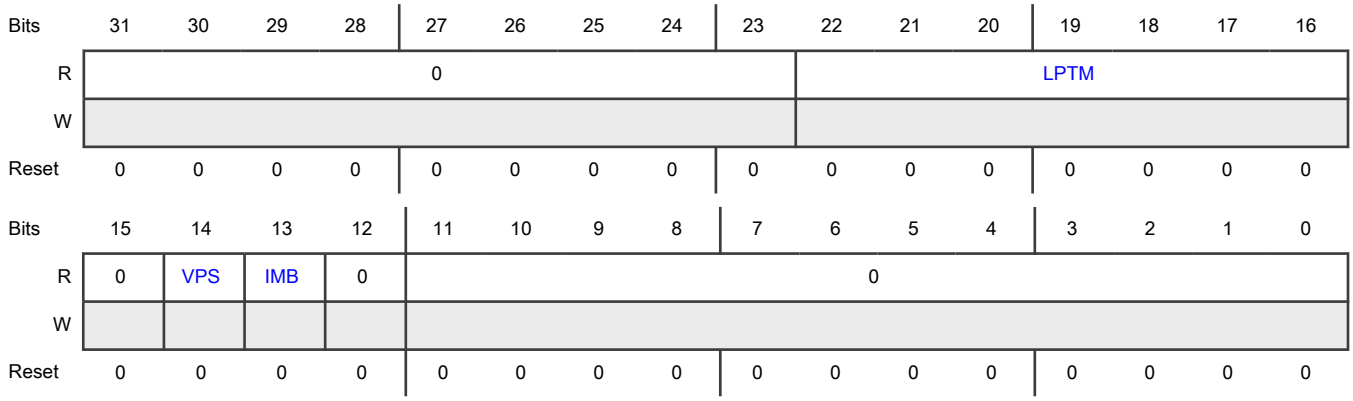
68.6.2.13 Error and Status 2 (ESR2)

Offset

Register	Offset
ESR2	38h

Function
Reports general status information.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-16 LPTM	Lowest Priority TX Message Buffer Indicates the lowest number inactive message buffer when ESR2[VPS] = 1 (see ESR2[IMB]). If no message buffer is inactive, the message buffer indicated depends on the value of CTRL1[LBUF]. If CTRL1[LBUF] = 0, the message buffer indicated is the one with the greatest arbitration value (see Highest-priority message buffer first). If CTRL1[LBUF] = 1, the message buffer indicated is the active TX message buffer with the highest number. If a TX message buffer is being transmitted, it is not considered in the LPTM calculation. If ESR2[IMB] is not 0 and a frame is transmitted successfully, the value of LPTM is updated with its message buffer number.
15 —	Reserved
14 VPS	Valid Priority Status Indicates whether the contents of ESR2[IMB] and ESR2[LPTM] are valid. It becomes 1 upon every complete TX arbitration process, unless the CPU writes to the Control and Status word of a message buffer already scanned. In other words, it is behind the TX Arbitration Pointer, during the TX arbitration process. If there is no inactive message buffer and only one TX message buffer that is being transmitted, this field remains 0. This field becomes 0 upon the start of every TX arbitration process or upon a write to the Control and Status word of any message buffer. **NOTE** No CPU write to the Control and Status of a message buffer that the abort mechanism blocks affects this field. When MCR[AEN] = 1, the abort code write to the Control and Status of an MB being transmitted (pending abort) is blocked. Any write attempt to a TX MB with its IFLAG flag set is also blocked. 0b - Invalid 1b - Valid

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 IMB	Inactive Message Buffer Indicates whether any message buffer is inactive (CODE field is either 1000b or 0b) when ESR2[VPS] = 1. This field becomes 1 when: • A lowest-priority TX message buffer (ESR2[LPTM]) is found and it is inactive during arbitration. • This field is not 1, and a frame is transmitted successfully. This field always becomes 0 at the start of arbitration (see Arbitration process). If a message buffer is successfully transmitted and this field is 0 (no inactive message buffer), ESR2[VPS] and this field both become 1. The index related to the MB transmitted is loaded into ESR2[LPTM]. In this case, the value of ESR2[LPTM] is the number of the first inactive message buffer. 0b - Message buffer indicated by ESR2[LPTM] is not inactive. 1b - At least one message buffer is inactive.
12 —	Reserved
11-0 —	Reserved

68.6.2.14 Cyclic Redundancy Check (CRCR)

Offset

Register	Offset
CRCR	44h

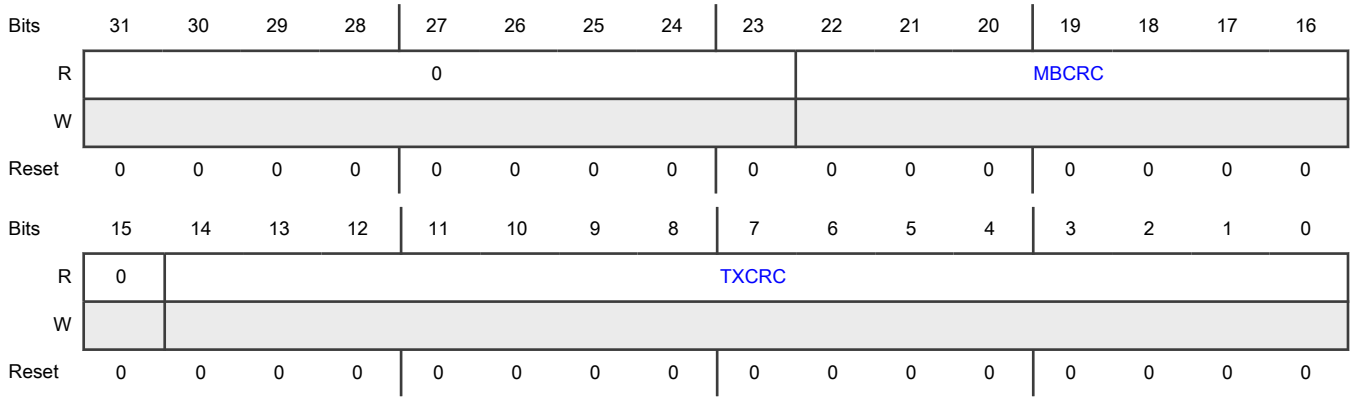
Function

Provides information about the CRC of transmitted messages for non-FD messages. This register only reports the 15 low-order bits of CRC calculations for messages in CAN FD format that require either 17 or 21 bits. For CAN FD format frames, you must use the CAN FD CRC (FDCRC). This register is updated at the same time that the TX interrupt flag is set.

NOTE

See CRC sequence calculation in the CAN Protocol standard (ISO 11898-1:2015) for details.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-16 MBCRC	CRC Message Buffer Indicates the number of the message buffer corresponding to the value in CRCR[TXCRC].
15 —	Reserved
14-0 TXCRC	Transmitted CRC value Indicates the CRC value of the last transmitted message for non-FD frames. For FD frames, CRC value is reported in CAN FD CRC (FDCRC).

68.6.2.15 Legacy RX FIFO Global Mask (RXFGMASK)

Offset

Register	Offset
RXFGMASK	48h

Function

Masks the Legacy RX FIFO ID filter table elements that do not have a corresponding RXIMR according to CTRL2[RFFN], when Legacy RX FIFO is enabled.

This register is located in RAM.

You can only write to this register in Freeze mode; the module blocks it in other modes.

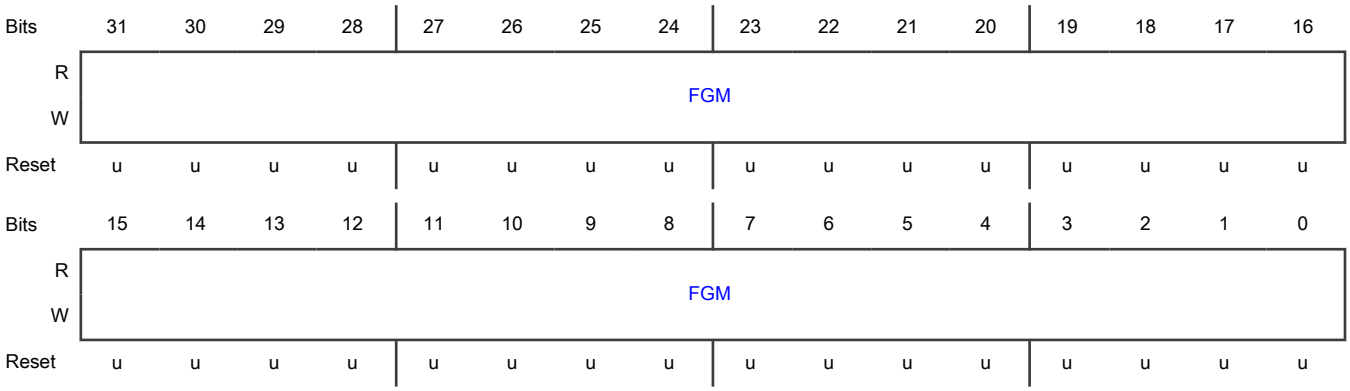
The following table shows how the FGM bits correspond to each IDAF field.

Table 612. Correspondence of Legacy RX FIFO global mask bits to IDF fields

Legacy RX FIFO ID filter table elements format (MCR[IDAM])	Identifier acceptance filter fields					
	RTR	IDE	RXIDA	RXIDB [1]	RXIDC [2]	Reserved
A	FGM[31]	FGM[30]	FGM[29:1]	—	—	FGM[0]
B	FGM[31], FGM[15]	FGM[30], FGM[14]	—	FGM[29:16], FGM[13:0]		—
C	—	—		—	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]	

1. If MCR[IDAM] is equivalent to format B, only the 14 most significant bits of the identifier of the incoming frame are compared with the Legacy RX FIFO filter.
2. If MCR[IDAM] is equivalent to format C, only the eight most significant bits of the identifier of the incoming frame are compared with the Legacy RX FIFO filter.

Diagram



Fields

Field	Function
31-0 FGM	Legacy RX FIFO Global Mask Bits Masks the ID filter table elements bits in a perfect alignment. 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

68.6.2.16 Legacy RX FIFO Information (RXFIR)

Offset

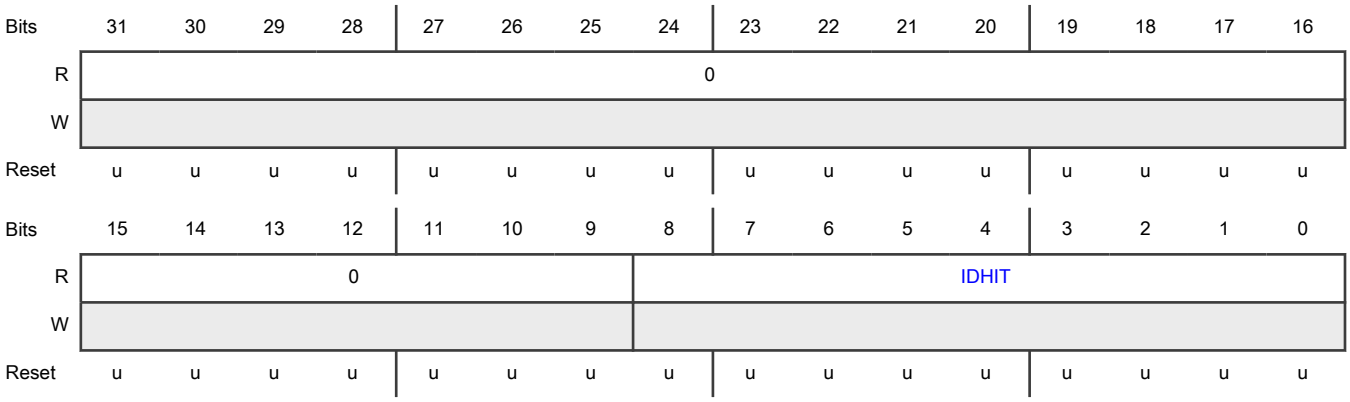
Register	Offset
RXFIR	4Ch

Function

Provides information about Legacy RX FIFO.

This register is the port through which the CPU accesses the output of the Legacy RXFIR FIFO located in RAM. FlexCAN writes to the Legacy RXFIR FIFO when a new message is moved into the Legacy RX FIFO. Also, its output is updated whenever the output of the Legacy RX FIFO is updated with the next message. See Legacy RX FIFO for instructions on reading this register.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-0 IDHIT	Identifier Acceptance Filter Hit Indicator Indicates which Identifier Acceptance filter that the received message hit in the output of the Legacy RX FIFO. If multiple filters match the incoming message ID, the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only when IFLAG1[BUF5I] is set.

68.6.2.17 CAN Bit Timing (CBT)

Offset

Register	Offset
CBT	50h

Function

Provides an alternative way to store the CAN bit timing variables described in Control 1 (CTRL1). EPRESDIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW are extended versions of CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] respectively.

NOTE

The CAN bit variables in CTRL1 and in CBT are stored in the same register.

CBT[BTF] selects the use of the timing variables defined in this register.

When the CAN FD feature is enabled (MCR[FDEN] = 1), always write 1 to CBT[BTF].

Soft reset does not affect the contents of this register.

NOTE

Ensure that bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1:2015).

NOTE

If CTRL2[BTE] = 1, EPRESDIV, ERJW, EPROPSEG, EPSEG1, and EPSEG2 are read as zero. A write operation to them has no effect.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BTF	EPRESDIV											ERJW			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EPROPSEG				EPSEG1				EPSEG2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 BTF	Bit Timing Format Enable Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW. These fields replace the CAN bit timing variables defined in Control 1 (CTRL1). This field can be written in Freeze mode only. 0b - Disable 1b - Enable
30-21 EPRESDIV	Extended Prescaler Division Factor Defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PRESDIV] value range. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). This field can be written only in Freeze mode; the module blocks it in other modes. Sclock frequency = PE clock frequency ÷ (EPRESDIV + 1)
20-16 ERJW	Extended Resync Jump Width Defines the maximum number of time quanta that one resynchronization can change a bit time when CBT[BTF] = 1. Otherwise, it has no effect. It extends the CTRL1[RJW] value range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field can be written only in Freeze mode; the module blocks it in other modes. Resync Jump Width = ERJW + 1. One Time Quantum = one Sclock period.
15-10 EPROPSEG	Extended Propagation Segment Defines the length of the propagation segment in the bit time when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PROPSEG] value range. This field can be written only in Freeze mode; the module blocks it in other modes. Propagation Segment Time = (EPROPSEG + 1) × Time Quanta. One Time Quantum = one Sclock period.
9-5 EPSEG1	Extended Phase Segment 1 Defines the length of phase segment 1 in the bit time when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PSEG1] value range. This field can be written only in Freeze mode; the module blocks it in other modes. Phase Buffer Segment 1 = (EPSEG1 + 1) × Time Quanta. One Time Quantum = one Sclock period.
4-0 EPSEG2	Extended Phase Segment 2 Defines the length of phase segment 2 in the bit time when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PSEG2] value range. This field can be written only in Freeze mode; the module blocks it in other modes. Phase Buffer Segment 1 = (EPSEG2 + 1) × Time Quanta. One Time Quantum = one Sclock period.

68.6.2.18 Receive Individual Mask (RXIMR0 - RXIMR31)

Offset

For n = 0 to 31:

Register	Offset
RXIMRn	880h + (n × 4h)

Function

Stores the acceptance masks for ID filtering in RX message buffers and the Legacy RX FIFO.

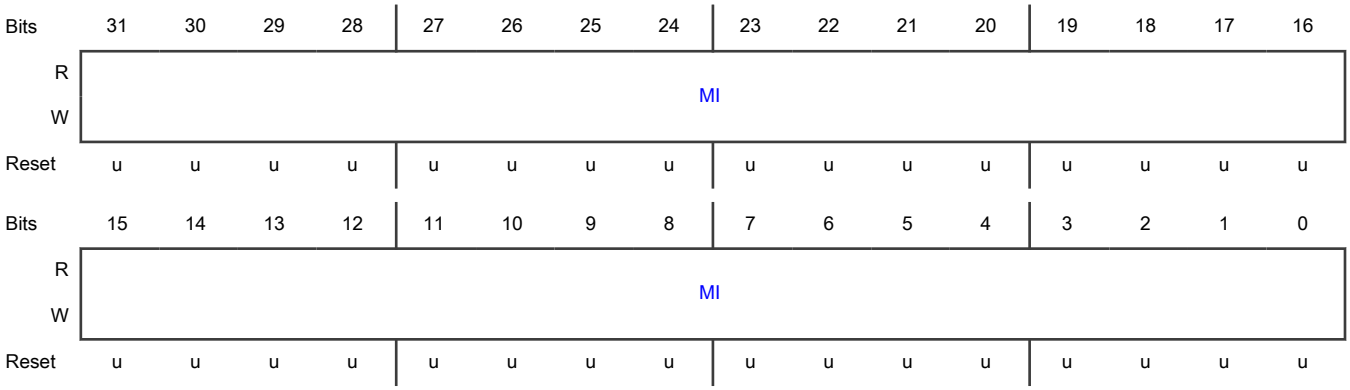
When the Legacy RX FIFO is disabled (MCR[RFEN] = 0), an individual mask is provided for each available RX message buffer on a one-to-one correspondence. When the Legacy RX FIFO is enabled (MCR[RFEN] = 1), an individual mask is provided for each Legacy RX FIFO ID filter table element on a one-to-one correspondence. This correspondence depends on the setting of CTRL2[RFFN] (see Legacy RX FIFO).

RXIMR0 stores the individual mask associated with either MB0 or ID filter table element 0. RXIMR1 stores the individual mask associated with either MB1 or ID filter table element 1, and so on.

The CPU can only access the RXIMR registers when the module is in Freeze mode; otherwise, the module blocks them. Reset does not affect these registers. They are located in RAM and must be explicitly initialized prior to any reception.

It is possible for the RXIMR memory region to be accessed as general-purpose memory. See Bus interface for more information.

Diagram



Fields

Field	Function
31-0 MI	Individual Mask Bits Masks the corresponding bit in both the message buffer filter and Legacy RX FIFO ID filter table element in distinct ways. For message buffer filters, see RX Message Buffers Global Mask (RXMGMASK). For Legacy RX FIFO ID filter table elements, see Legacy RX FIFO Global Mask (RXFGMASK). 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

68.6.2.19 Pretended Networking Control 1 (CTRL1_PN)

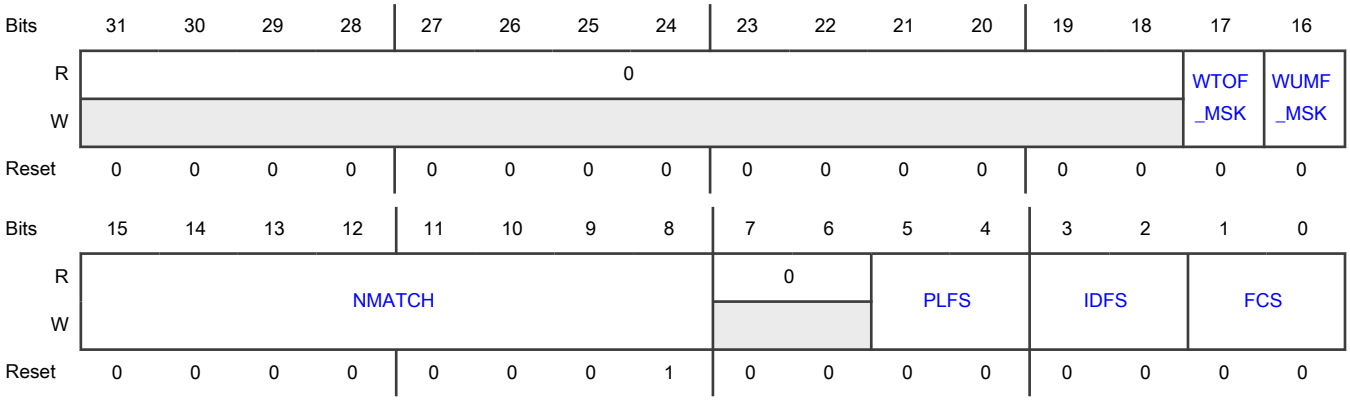
Offset

Register	Offset
CTRL1_PN	B00h

Function

Contains control bits for Pretended Networking mode filtering selection. Configure this register with the filter criteria to be used to receive wake-up messages. Fields in this register can be written in Freeze mode only, except for CTRL1_PN[WTOF_MSK] and CTRL1_PN[WUMF_MSK].

Diagram



Fields

Field	Function
31-18 —	Reserved
17 WTOF_MSK	Wake-up by Timeout Flag Mask Enables the generation of a wake-up event originated by a timeout. 0b - Disable 1b - Enable
16 WUMF_MSK	Wake-up by Matching Flag Mask Enables the generation of a wake-up event originated by a successful filtered RX message. 0b - Disable 1b - Enable
15-8 NMATCH	Number of Messages Matching the Same Filtering Criteria Defines the number of times a given message must match the predefined filtering criteria for ID or payload before generating a wake-up event. You can configure this quantity in the 1–255 range by using values 01h–FFh, respectively. 0000_0001b - Once 0000_0010b - Twice 1111_1111b - 255 times
7-6 —	Reserved
5-4 PLFS	Payload Filtering Selection Selects the level of payload filtering to be applied when FlexCAN is in Pretended Networking mode. When payload filtering is active, filtering does not accept remote messages (RTR = 1).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Match payload contents to an exact target value 01b - Match a payload value greater than or equal to a specified target value 10b - Match a payload value smaller than or equal to a specified target value 11b - Match upon a payload value within a range of values, inclusive
3-2 IDFS	ID Filtering Selection Selects the level of ID filtering to be applied when FlexCAN is in Pretended Networking mode. In ID filtering, if FLT_ID2_IDMASK[IDE_MSK] = 1 and FLT_ID2_IDMASK[RTR_MSK] = 1, the IDE and RTR bits are considered part of the reception filter. 00b - Match ID contents to an exact target value 01b - Match an ID value greater than or equal to a specified target value 10b - Match an ID value smaller than or equal to a specified target value 11b - Match an ID value within a range of values, inclusive
1-0 FCS	Filtering Combination Selection Selects the filtering criteria to be applied when FlexCAN is in Pretended Networking mode. See Receive process in Pretended Networking mode for more details. 00b - Message ID filtering only 01b - Message ID filtering and payload filtering 10b - Message ID filtering occurring a specified number of times 11b - Message ID filtering and payload filtering a specified number of times

68.6.2.20 Pretended Networking Control 2 (CTRL2_PN)

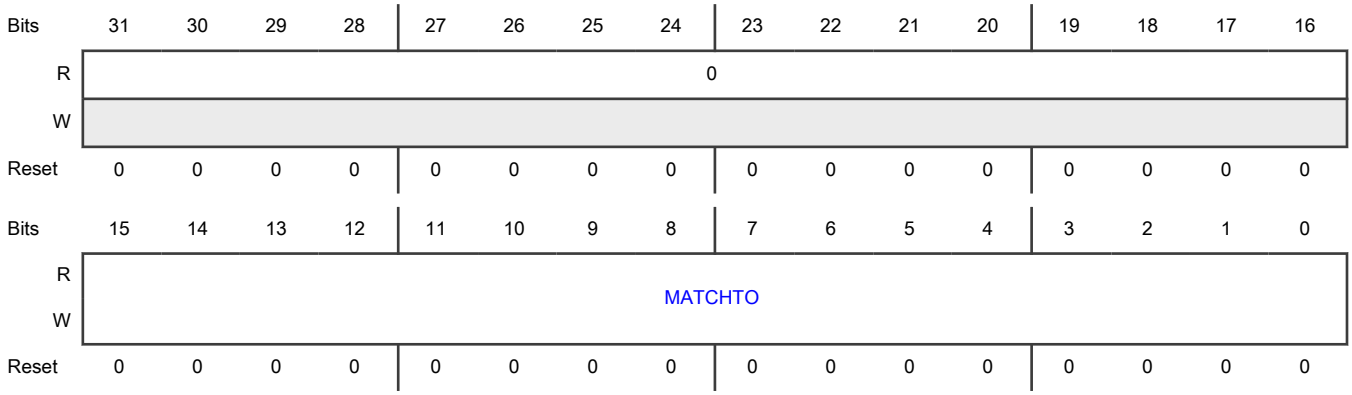
Offset

Register	Offset
CTRL2_PN	B04h

Function

Contains the configuration for the timeout value in Pretended Networking mode. You can only write to this register in Freeze mode.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 MATCHTO	Timeout for No Message Matching the Filtering Criteria Defines a timeout value that generates a wake-up event if MCR[PNET_EN] = 1. If the timeout counter reaches the target value when FlexCAN is in Pretended Networking mode, a wake-up event is generated. The timeout limit can be configured from 1 to 65535 to control an internal 16-bit incrementing timer to produce a trigger upon reaching this configured value. The internal timer is incremented based on periodic time ticks, where the period is 64 times the CAN Bit Time unit. Writing 0000h to this field disables the timeout.

68.6.2.21 Pretended Networking Wake-Up Match (WU_MTC)

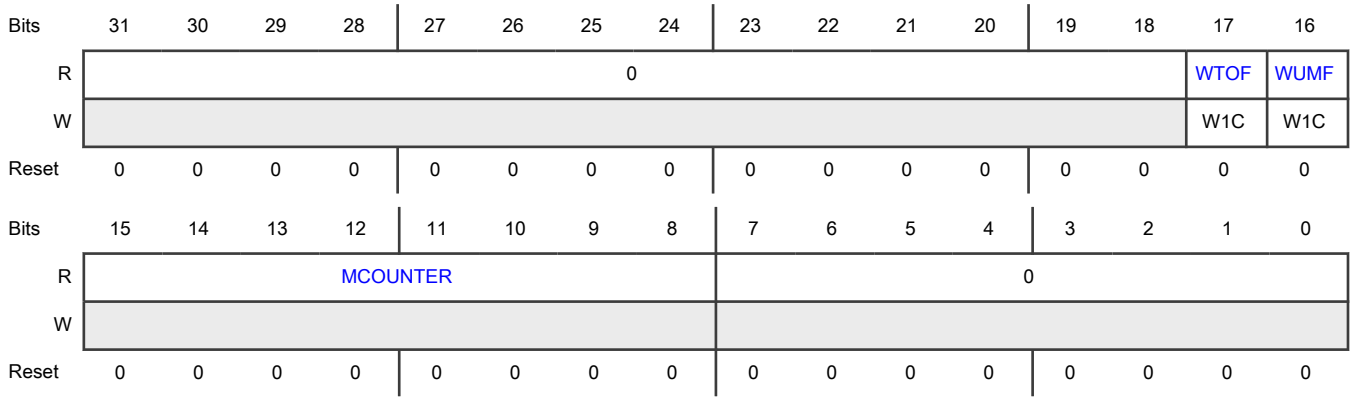
Offset

Register	Offset
WU_MTC	B08h

Function

Contains wake-up information related to the matching processes performed when FlexCAN receives frames in Pretended Networking mode.

Diagram



Fields

Field	Function
31-18 —	Reserved
17 WTOF	Wake-up by Timeout Flag Bit Identifies whether FlexCAN has detected a timeout event during a time interval defined by CTRL2_PN[MATCHTO]. If CTRL1_PN[WTOF_MSK] = 1, this flag generates a wake-up event. 0b - No event detected 1b - Event detected
16 WUMF	Wake-up by Match Flag Identifies whether FlexCAN has detected a matching RX incoming message that meets the filtering criteria specified in Pretended Networking Control 1 (CTRL1_PN). If CTRL1_PN[WUMF_MSK] = 1, this flag generates a wake-up event. 0b - No event detected 1b - Event detected
15-8 MCOUNTER	Number of Matches in Pretended Networking Contains the number of times a given message has matched the predefined filtering criteria for ID or payload before a wake-up event. When FlexCAN enters Pretended Networking mode, this register is reset; soft reset does not affect it.
7-0 —	Reserved

68.6.2.22 Pretended Networking ID Filter 1 (FLT_ID1)

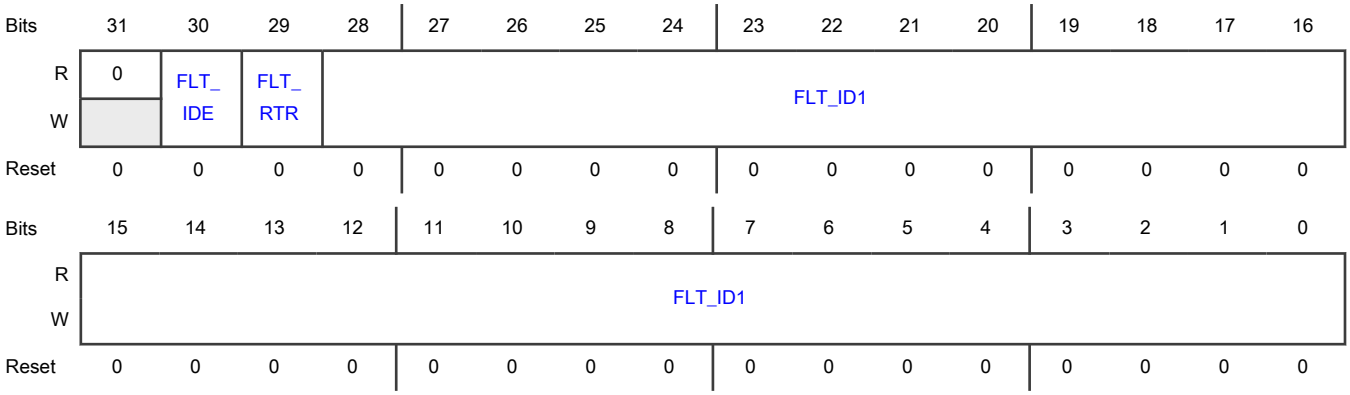
Offset

Register	Offset
FLT_ID1	B0Ch

Function

Contains FLT_ID1 target value, as well as IDE and RTR target values used to filter an incoming message ID. The FLT_ID1 is used for comparisons or as the lower limit value in an ID range detection. It can be written in Freeze mode only.

Diagram



Fields

Field	Function
31 —	Reserved
30 FLT_IDE	ID Extended Filter Identifies whether the frame format is standard or extended. It is used as part of the ID reception filter. 0b - Standard 1b - Extended
29 FLT_RTR	Remote Transmission Request Filter Identifies whether the frame is remote. It is used as part of the ID reception filter. 0b - Reject remote frame (accept data frame) 1b - Accept remote frame
28-0 FLT_ID1	ID Filter 1 for Pretended Networking filtering Defines either the 29 bits of an extended frame format, considering all bits, or the 11 bits of a standard frame format, considering only the leftmost 11 bits.

68.6.2.23 Pretended Networking Data Length Code (DLC) Filter (FLT_DLC)

Offset

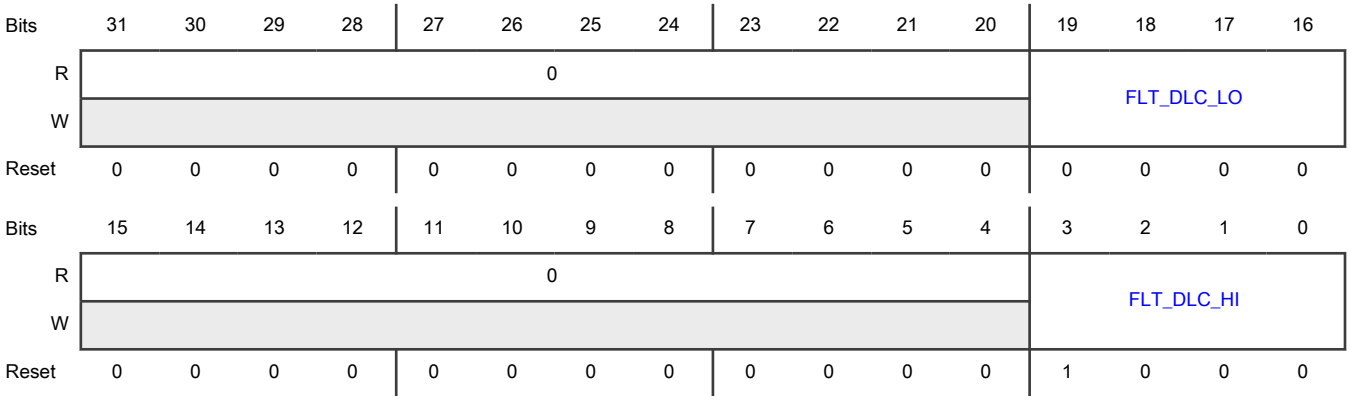
Register	Offset
FLT_DLC	B10h

Function

Contains the DLC inside a range of target values (FLT_DLC_LO and FLT_DLC_HI) used to filter an incoming message. The DLC range is used only for payload filtering. It can be written in Freeze mode only.

When a fixed quantity of data bytes is required, write the same value to FLT_DLC[FLT_DLC_LO] and FLT_DLC[FLT_DLC_HI]. See Receive process in Pretended Networking mode.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 FLT_DLC_LO	Lower Limit for Length of Data Bytes Filter Specifies the lower limit for the number of data bytes considered valid for payload comparison. It is used as part of payload reception filter.
15-4 —	Reserved
3-0 FLT_DLC_HI	Upper Limit for Length of Data Bytes Filter Specifies the upper limit for the number of data bytes considered valid for payload comparison. It is used as part of payload reception filter.

68.6.2.24 Pretended Networking Payload Low Filter 1 (PL1_LO)

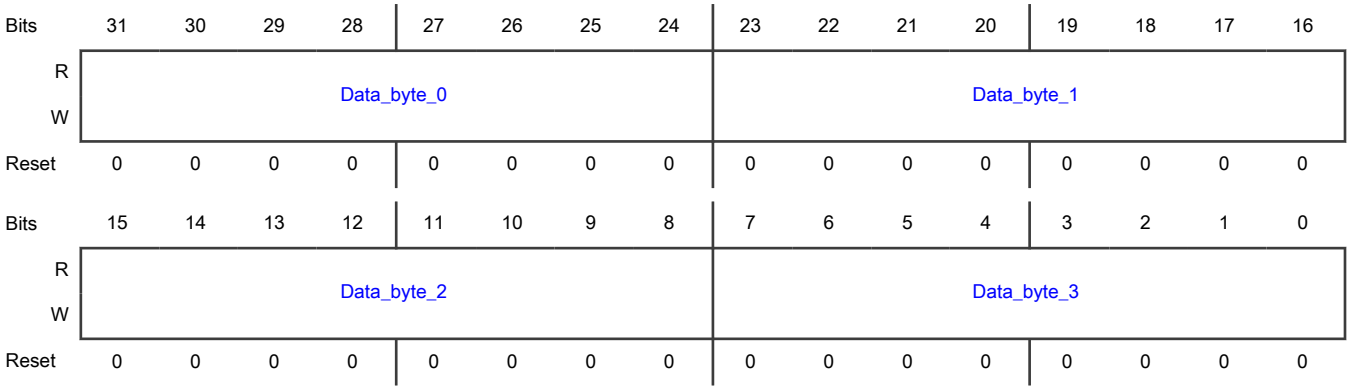
Offset

Register	Offset
PL1_LO	B14h

Function

Contains the low-order bits of the target value used to filter incoming message payload for payload filter 1. It is used for comparisons or as the lower limit value in a payload range detection. It can be written in Freeze mode only.

Diagram



Fields

Field	Function
31-24 Data_byte_0	Data byte 0 Contains payload filter 1 low-order bits for PN payload filtering corresponding to data byte 0.
23-16 Data_byte_1	Data byte 1 Contains payload filter 1 low-order bits for PN payload filtering corresponding to data byte 1.
15-8 Data_byte_2	Data byte 2 Contains payload filter 1 low-order bits for PN payload filtering corresponding to data byte 2.
7-0 Data_byte_3	Data byte 3 Contains payload filter 1 low-order bits for PN payload filtering corresponding to data byte 3.

68.6.2.25 Pretended Networking Payload High Filter 1 (PL1_HI)

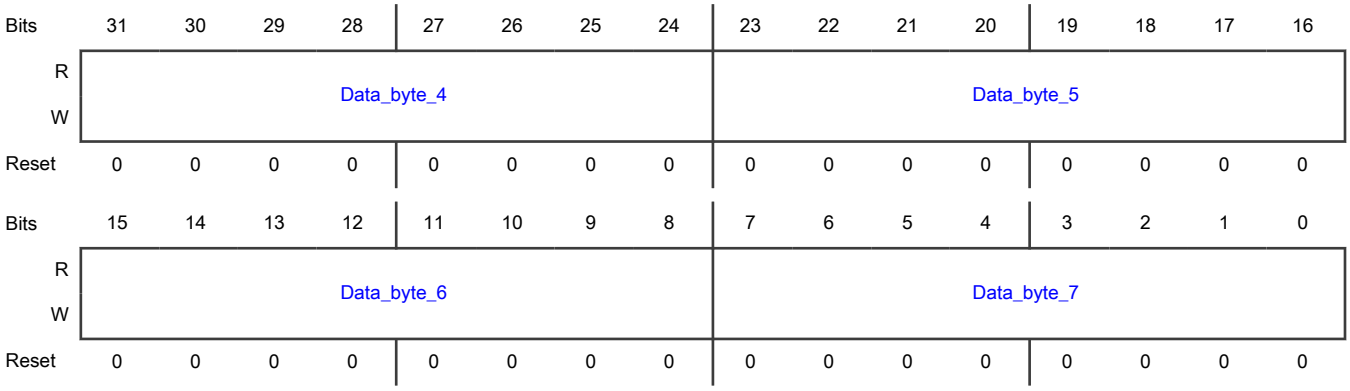
Offset

Register	Offset
PL1_HI	B18h

Function

Contains the high-order bits of the target value used to filter incoming message payload for payload filter 1. It is used either for comparisons or as the lower limit value in a payload range detection. It can be written in Freeze mode only.

Diagram



Fields

Field	Function
31-24 Data_byte_4	Data byte 4 Contains payload filter 1 high-order bits for PN payload filtering corresponding to data byte 4.
23-16 Data_byte_5	Data byte 5 Contains payload filter 1 high-order bits for PN payload filtering corresponding to data byte 5.
15-8 Data_byte_6	Data byte 6 Contains payload filter 1 high-order bits for PN payload filtering corresponding to data byte 6.
7-0 Data_byte_7	Data byte 7 Contains payload filter 1 high-order bits for PN payload filtering corresponding to data byte 7.

68.6.2.26 Pretended Networking ID Filter 2 or ID Mask (FLT_ID2_IDMASK)

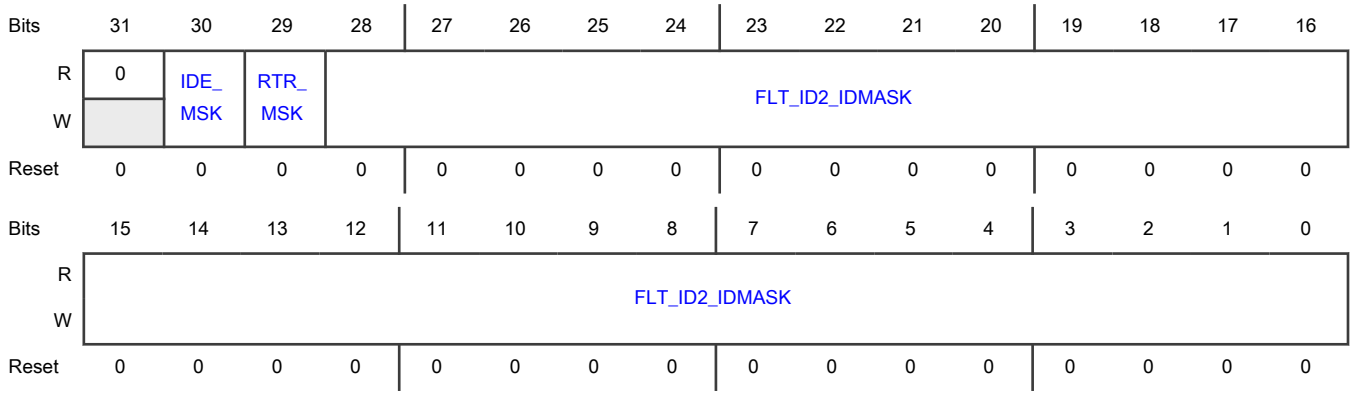
Offset

Register	Offset
FLT_ID2_IDMASK	B1Ch

Function

Contains FLT_ID2 target value used only as the upper limit value in ID range detection. When an exact ID filtering criterion is selected, this register stores the ID mask. IDE_MSK and RTR_MSK are used in both types of ID filtering (exact and range). These fields enable FLT_ID1[FLT_IDE] and FLT_ID1[FLT_RTR] to be used as part of the ID reception filter. This register can be written in Freeze mode only.

Diagram



Fields

Field	Function
31 —	Reserved
30 IDE_MSK	ID Extended Mask Indicates whether the frame format (standard or extended) is used as part of the ID reception filter. 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.
29 RTR_MSK	Remote Transmission Request Mask Indicates whether the frame type (data or remote) is part of the ID reception filter. 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.
28-0 FLT_ID2_IDMASK	ID Filter 2 for Pretended Networking Filtering or ID Mask Bits for Pretended Networking ID Filtering Defines filter values in range ID filtering: • Value in extended frame format (29 bits), considering FLT_ID2[28:0] • Value in standard frame format (11 bits), considering the FLT_ID2[28:18]. In this case, bits [17:0] are meaningless. Or, defines the mask values in exact ID filtering: • Values for extended frame format (29 bits), considering IDMASK[28:0] • Values for standard frame format (11 bits), considering IDMASK[28:18]. In this case, bits [17:0] are meaningless.

68.6.2.27 Pretended Networking Payload Low Filter 2 and Payload Low Mask (PL2_PLMASK_LO)

Offset

Register	Offset
PL2_PLMASK_LO	B20h

Function

Contains the low-order bits for Payload Filter 2, used only as the upper limit value in a payload range detection. Also, when an exact payload filtering criterion is selected, this register is used as payload mask for the low-order bits. Otherwise, this register is unused. It can be written in Freeze mode only.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Data_byte_0								Data_byte_1							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Data_byte_2								Data_byte_3							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 Data_byte_0	Data Byte 0 Contains payload filter 2 low-order bits, or Payload Mask low-order bits for PN payload filtering, corresponding to data byte 0
23-16 Data_byte_1	Data Byte 1 Contains payload filter 2 low-order bits, or Payload Mask low-order bits for PN payload filtering, corresponding to data byte 1
15-8 Data_byte_2	Data Byte 2 Contains payload filter 2 low-order bits, or Payload Mask low-order bits for PN payload filtering, corresponding to data byte 2
7-0 Data_byte_3	Data Byte 3 Contains payload filter 2 low-order bits, or Payload Mask low-order bits for PN payload filtering, corresponding to data byte 3

68.6.2.28 Pretended Networking Payload High Filter 2 and Payload High Mask (PL2_PLMASK_HI)

Offset

Register	Offset
PL2_PLMASK_HI	B24h

Function

Contains the high-order bits for the Payload Filter 2, used only as the upper limit value in a payload range detection. Also, when an exact payload filtering criterion is selected, this register is used as payload mask for the high-order bits. Otherwise, this register is unused. It can be written in Freeze mode only.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Data_byte_4								Data_byte_5							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Data_byte_6								Data_byte_7							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 Data_byte_4	Data Byte 4 Contains payload filter 2 high-order bits, or Payload Mask high-order bits for PN payload filtering, corresponding to data byte 4.
23-16 Data_byte_5	Data Byte 5 Contains payload filter 2 high-order bits, or Payload Mask high-order bits for PN payload filtering, corresponding to data byte 5.
15-8 Data_byte_6	Data Byte 6 Contains payload filter 2 high-order bits, or Payload Mask high-order bits for PN payload filtering, corresponding to data byte 6.
7-0 Data_byte_7	Data Byte 7 Contains payload filter 2 high-order bits, or Payload Mask high-order bits for PN payload filtering, corresponding to data byte 7.

68.6.2.29 Wake-Up Message Buffer (WMB0_CS - WMB3_CS)

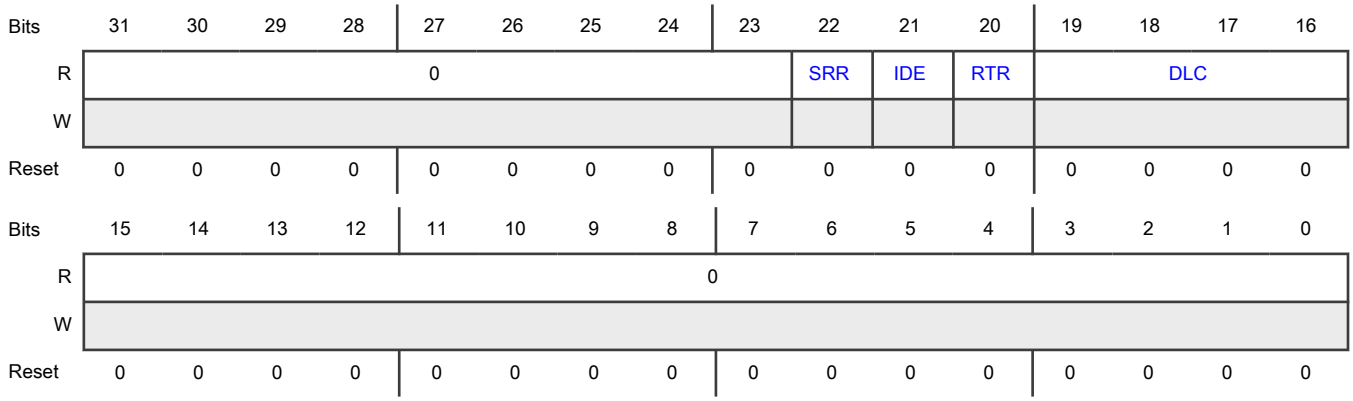
Offset

Register	Offset
WMB0_CS	B40h
WMB1_CS	B50h
WMB2_CS	B60h
WMB3_CS	B70h

Function

Stores the Control and Status information (IDE, RTR, and DLC fields) of an incoming RX message.

Diagram



Fields

Field	Function
31-23 —	Reserved
22 SRR	Substitute Remote Request Identifies whether the request is dominant or recessive. This field is used only in extended format. You must write 1 to this field for transmission buffers. The value of this field is stored with the value received on the CAN bus for receiving buffers. If FlexCAN receives this field as dominant, it is interpreted as an arbitration loss. 0b - Dominant 1b - Recessive
21 IDE	ID Extended Bit Identifies whether the frame format is standard or extended.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Standard 1b - Extended
20 RTR	Remote Transmission Request Identifies whether the frame is remote or data 0b - Data 1b - Remote
19-16 DLC	Length of Data in Bytes Contains the length (in bytes) of the RX data received when FlexCAN is in Pretended Networking mode. FlexCAN writes this field, copied from the DLC (Data Length Code) field of the received frame. The DLC field indicates which data bytes are valid.
15-0 —	Reserved

68.6.2.30 Wake-Up Message Buffer for ID (WMB0_ID - WMB3_ID)

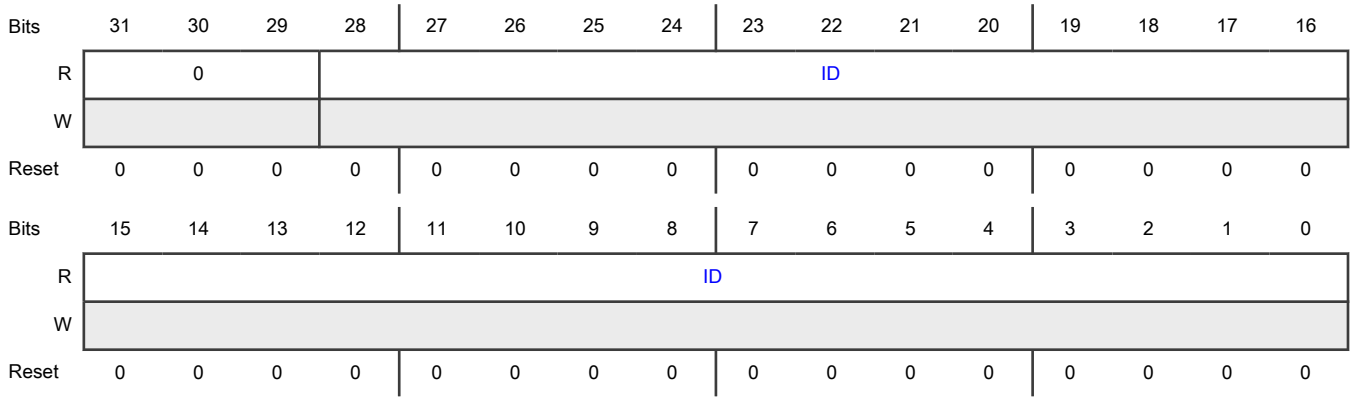
Offset

Register	Offset
WMB0_ID	B44h
WMB1_ID	B54h
WMB2_ID	B64h
WMB3_ID	B74h

Function

Stores the ID information of an incoming RX message.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-0 ID	Received ID in Pretended Networking Mode Stores the received ID, which is: • The 29 bits of the extended frame format (considering ID[28:0]) • The 11 bits of the standard frame format (considering ID[28:18] only; the remaining bits in the ID[17:0] range are meaningless).

68.6.2.31 Wake-Up Message Buffer for Data 0–3 (WMB0_D03 - WMB3_D03)

Offset

Register	Offset
WMB0_D03	B48h
WMB1_D03	B58h
WMB2_D03	B68h
WMB3_D03	B78h

Function

Stores data bytes 0–3 of the payload information of an incoming RX message. The content of each register is cleared when the incoming matched message is either a remote frame (RTR = 1) or a data frame with DLC = 0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Data_byte_0								Data_byte_1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Data_byte_2								Data_byte_3							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 Data_byte_0	Data Byte 0 Contains received payload corresponding to data byte 0 in Pretended Networking mode
23-16 Data_byte_1	Data Byte 1 Contains received payload corresponding to data byte 1 in Pretended Networking mode
15-8 Data_byte_2	Data Byte 2 Contains received payload corresponding to data byte 2 in Pretended Networking mode
7-0 Data_byte_3	Data Byte 3 Contains received payload corresponding to data byte 3 in Pretended Networking mode

68.6.2.32 Wake-Up Message Buffer Register Data 4–7 (WMB0_D47 - WMB3_D47)

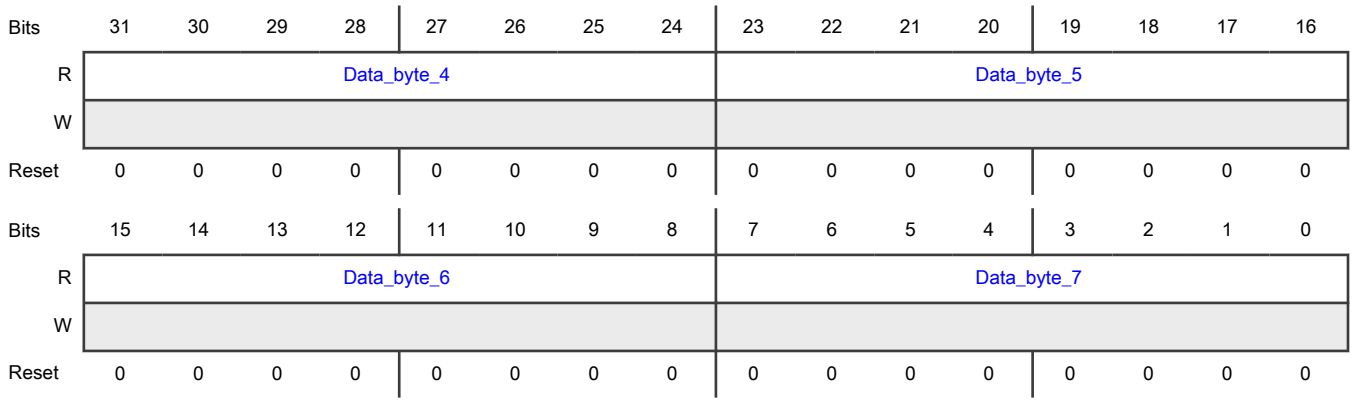
Offset

Register	Offset
WMB0_D47	B4Ch
WMB1_D47	B5Ch
WMB2_D47	B6Ch
WMB3_D47	B7Ch

Function

Stores the data bytes 4–7 of the payload information of an incoming RX message. The content of each register is cleared when the incoming matched message is either a remote frame (RTR = 1) or a data frame with DLC = 0.

Diagram



Fields

Field	Function
31-24 Data_byte_4	Data Byte 4 Contains received payload corresponding to data byte 4 in Pretended Networking mode
23-16 Data_byte_5	Data Byte 5 Contains received payload corresponding to data byte 5 in Pretended Networking mode
15-8 Data_byte_6	Data Byte 6 Contains received payload corresponding to data byte 6 in Pretended Networking mode
7-0 Data_byte_7	Data Byte 7 Contains received payload corresponding to data byte 7 in Pretended Networking mode

68.6.2.33 Enhanced CAN Bit Timing Prescalers (EPRS)

Offset

Register	Offset
EPRS	BF0h

Function

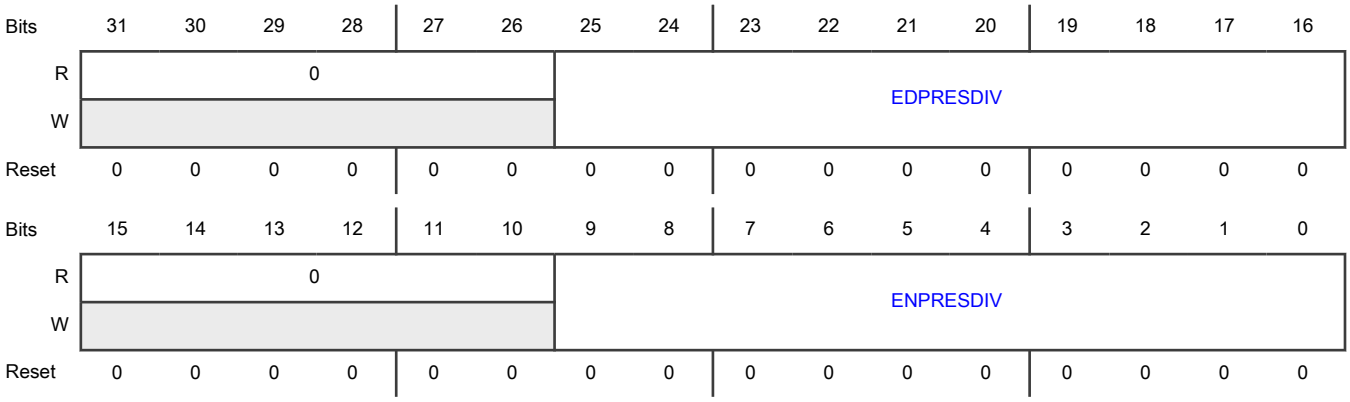
Defines the CAN bit timing prescalers for the nominal phase and data phase when CTRL2[BTE] = 1.

Used by the module only if CTRL2[BTE] = 1; otherwise a write operation has no effect and all fields are read as zero.

This register can be written only in Freeze mode; the module blocks it in other modes.

Soft reset does not affect the contents of this register.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 EDPRESDIV	Extended Data Phase Prescaler Division Factor Defines the ratio between the PE clock frequency and the Sclock frequency in the data phase of a CAN FD message when CTRL2[BTE] = 1. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. Sclock frequency = PE clock frequency ÷ (EDPRESDIV + 1) **NOTE** To minimize errors when processing FD frames, use the same value for this field and for EPRS[ENPRESDIV]. See the first note in CAN FD frames for details.
15-10 —	Reserved
9-0 ENPRESDIV	Extended Nominal Prescaler Division Factor Defines the ratio between the PE clock frequency and the Sclock frequency when CTRL2[BTE] = 1. Otherwise, it reads as 0 and a write operation has no effect. The Sclock period defines the time quantum of the CAN protocol in the nominal phase. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). Sclock frequency = PE clock frequency ÷ (ENPRESDIV + 1)

68.6.2.34 Enhanced Nominal CAN Bit Timing (ENCBT)

Offset

Register	Offset
ENCBT	BF4h

Function

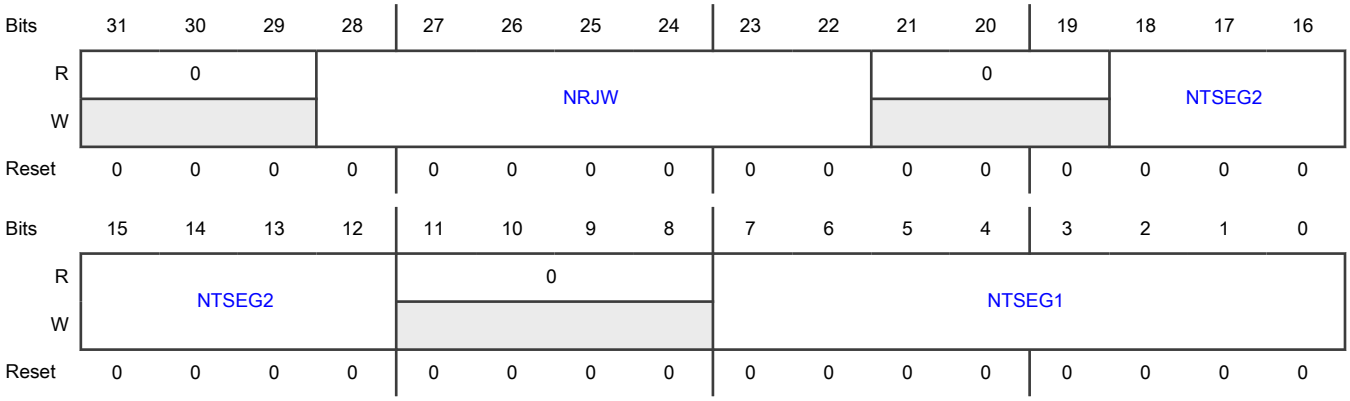
Provides an alternative way to store the CAN bit timing variables described in Control 1 (CTRL1) and CAN Bit Timing (CBT), to get higher CAN bit timing resolution.

This register is used by the module only if CTRL2[BTE] = 1. Otherwise, a write operation has no effect and all fields are read as zero.

Soft reset does not affect the contents of this register.

This register can be written only in Freeze mode; the module blocks it in other modes.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-22 NRJW	Nominal Resynchronization Jump Width Defines the maximum number of time quanta that one resynchronization can change a nominal bit time when CTRL2[BTE] = 1. Otherwise, it has no effect. One time quantum = one Sclock period Nominal Resync Jump Width = NRJW + 1
21-19 —	Reserved
18-12 NTSEG2	Nominal Time Segment 2 Defines the length of Time Segment 2 in the nominal bit time when CTRL2[BTE] = 1. Otherwise, it has no effect. Nominal Time Segment 2 = (NTSEG2 + 1) × Time Quanta One time quantum = one Sclock period
11-8 —	Reserved
7-0 NTSEG1	Nominal Time Segment 1 Defines the length of Time Segment 1 in the bit time when CTRL2[BTE] = 1. Otherwise, it has no effect. Nominal Time Segment 1 = (NTSEG1 + 1) × Time Quanta One time quantum = one Sclock period

68.6.2.35 Enhanced Data Phase CAN Bit Timing (EDCBT)

Offset

Register	Offset
EDCBT	BF8h

Function

Provides an alternative way to store the data phase CAN bit timing variables described in CAN FD Bit Timing (FDCBT) to achieve higher CAN bit timing resolution.

This register is used by the module only if CTRL2[BTE] = 1; otherwise a write operation has no effect and all fields are read as zero.

Soft reset does not affect the contents of this register.

This register can be written only in Freeze mode; the module blocks it in other modes.

NOTE

Ensure that bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1:2015).

NOTE

DTSEG1 must be at least two time quanta.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						DRJW				0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DTSEG2				0							DTSEG1				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25-22 DRJW	Data Phase Resynchronization Jump Width Defines the maximum number of time quanta that one resynchronization can change a data phase bit time when CTRL2[BTE] = 1. Otherwise, it has no effect. Data Phase Resync Jump Width = DRJW + 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
21-16 —	Reserved
15-12 DTSEG2	Data Phase Time Segment 2 Defines the length of time segment 2 in the data phase bit time when CTRL2[BTE] = 1. Otherwise, it has no effect. Data Phase Time Segment 2 = (DTSEG2 + 1) × Time Quanta. One Time Quantum = one Sclock period.
11-5 —	Reserved
4-0 DTSEG1	Data Phase Segment 1 Defines the length of time segment 1 in the data phase bit time when CTRL2[BTE] = 1. Otherwise, it has no effect. Data Phase Time Segment 1 = (NTSEG1 + 1) × Time Quanta. One Time Quantum = one Sclock period.

68.6.2.36 Enhanced Transceiver Delay Compensation (ETDC)

Offset

Register	Offset
ETDC	BFCh

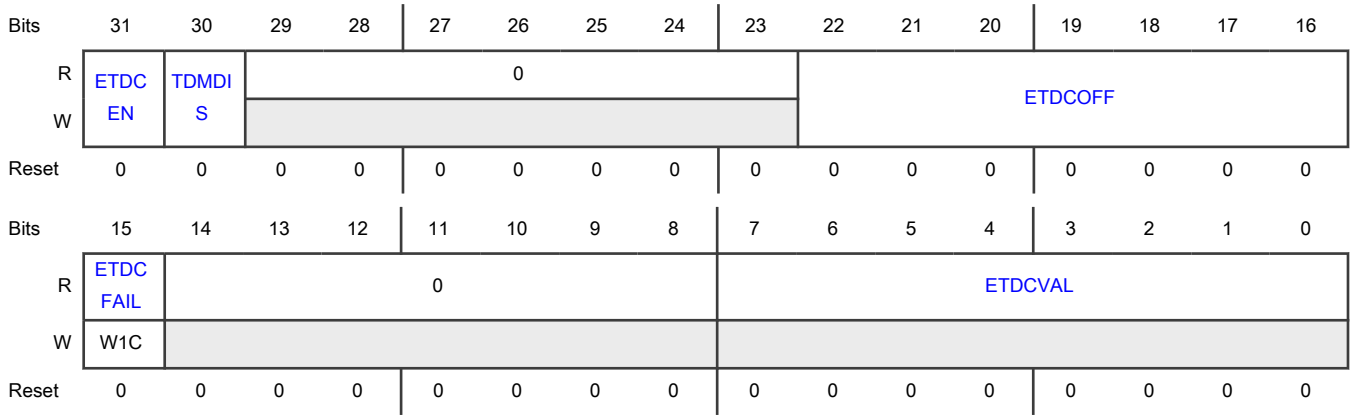
Function

Contains extended versions of FDCTRL[TDCOFF] and FDCTRL[TDCVAL]. This register is used by the module only if CTRL2[BTE] = 1. Otherwise, a write operation has no effect and all fields are read as zero.

NOTE

See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1:2015) for details.

Diagram



Fields

Field	Function
31 ETDCEN	Transceiver Delay Compensation Enable Enables the TDC feature. It can be written in Freeze mode only. **NOTE** See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1:2015) for details. **NOTE** TDC must be disabled when the Loop Back Mode is enabled. See CTRL1[LPB]. 0b - Disable 1b - Enable
30 TDMDIS	Transceiver Delay Measurement Disable Disables the transceiver delay measurement. When the TDC measurement is disabled, only ETDC[ETDCOFF] determines the secondary sample point position. If TCD measurement is enabled, the sum of the transceiver delay measurement plus the enhanced TDC offset determines the secondary sample point position. Soft reset does not affect this field. **NOTE** This bit can be enabled only if CTRL2[BTE] = 1. 0b - Enable 1b - Disable
29-23 —	Reserved
22-16	Enhanced Transceiver Delay Compensation Offset

Table continues on the next page...

Table continued from the previous page...

Field	Function
ETDCOFF	Contains the offset value to be added to the loop delay of the measured transceiver. This value defines the position of the delayed comparison point when bit rate switching is active. See Transceiver delay compensation for details on how the loop delay measurement is performed. This field can be written in Freeze mode only. Its value can be defined in protocol engine (PE) clock periods (CANCLK, see Protocol timing for more details). It must be smaller than the CAN bit duration in the data bit rate for proper operation. Do not write 0 to this field. **NOTE** If CTRL2[BTE] becomes 1 after a chip-level hard reset, this field is read as 1h.
15 ETDCFAIL	Transceiver Delay Compensation Fail Indicates whether the transceiver delay compensation (TDC) mechanism is out of range. In this case, it is unable to compensate the loop delay of the transceiver and successfully compare the delayed received bits to the transmitted ones. (See Transceiver delay compensation.) This field becomes 0 the first time FlexCAN detects the out of range condition. 0b - In range 1b - Out of range
14-8 —	Reserved
7-0 ETDCVAL	Enhanced Transceiver Delay Compensation Value Contains ETDC[ETDCOFF] added to the measured value of the transceiver loop delay in the latest transmitted CAN FD frame, with BRS = 1. The module only updates this field when ETDC[ETDCEN] = 1. Soft reset affects this field. **NOTE** If ETDC[TDMDIS] = 1, this field stores ETDC[ETDCOFF] only.

68.6.2.37 CAN FD Control (FDCTRL)

Offset

Register	Offset
FDCTRL	C00h

Function

Contains control bits for CAN FD operation. It also defines the data size of message buffers allocated in different partitions of RAM (memory blocks) as described in Table 613.

When an 8-byte payload is selected:

- Block R0 allocates MB0–MB31.
- Block R1 allocates MB32–MB63.

When a payload larger than eight bytes is selected, the maximum number of message buffers in a block is limited as described below.

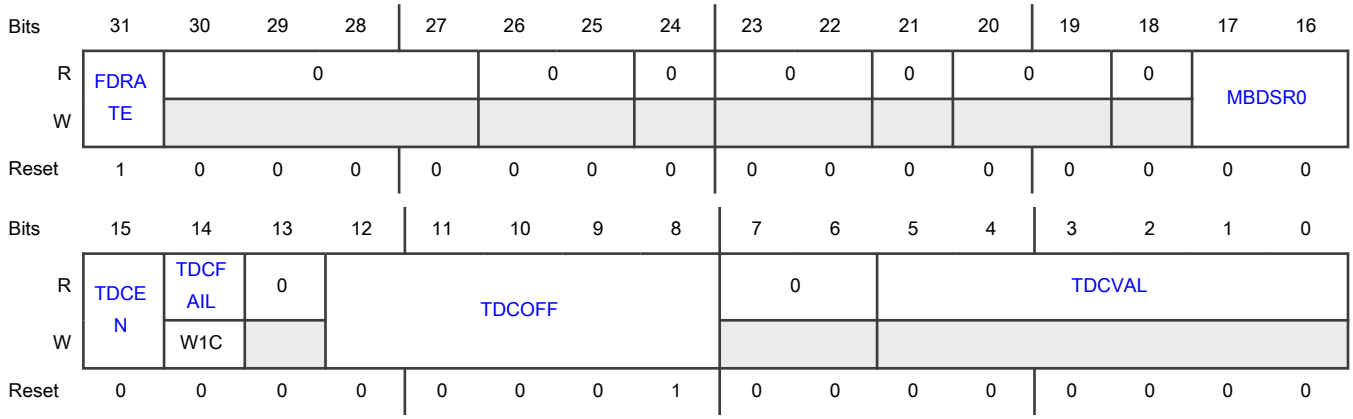
Table 613. Number of message buffers

Payload size	Maximum number of message buffers per RAM block
8 bytes	32
16 bytes	21
32 bytes	12
64 bytes	7

One memory block fits exactly 32 message buffers with an 8-byte payload. For other possible payload sizes, empty memory may exist between the last message buffer in a block and the beginning of the next block. This empty memory corresponds to less than one message buffer, and must not be used.

Soft reset does not affect the contents of this register.

Diagram



Fields

Field	Function
31 FDRATE	Bit Rate Switch Enable Enables the effect of the Bit Rate Switch (BRS bit) during the data phase of TX messages. When 1, if BRS = 1 in the TX message buffer, frames are transmitted with bit rate switching. When 0, frames are transmitted at a nominal rate, and the BRS bit in the TX MB has no effect. The CPU can write to this field at any time. However, its effect becomes active only under one of these conditions: • The CAN bus is in the Wait for Bus Idle state. • The CAN bus is in the Bus Idle state. • The CAN bus is in the Bus Off state.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• The current frame under reception or transmission reaches the interframe space. By writing 0 to FDCTRL[FDRATE], the CPU can force all bits in CAN FD messages to be transmitted at nominal bit rate. This transmission occurs regardless of the value in the BRS bit of the TX message buffers. 0b - Disable 1b - Enable
30-27 —	Reserved
26-25 —	Reserved
24 —	Reserved
23-22 —	Reserved
21 —	Reserved
20-19 —	Reserved
18 —	Reserved
17-16 MBDSR0	Message Buffer Data Size for Region 0 Selects the data size per message buffer for region R0 of message buffers allocated in RAM. This field can be written in Freeze mode only. 00b - 8 bytes 01b - 16 bytes 10b - 32 bytes 11b - 64 bytes
15 TDCEN	Transceiver Delay Compensation Enable Enables the TDC feature. It can be written in Freeze mode only. See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1:2015) for details. TDC must be disabled when Loopback mode is enabled (see CTRL1[LPB]).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE If CTRL2[BTE] = 1, this field is read as 0 and a write operation has no effect. 0b - Disable 1b - Enable
14 TDCFAIL	Transceiver Delay Compensation Fail Indicates whether the Transceiver Delay Compensation (TDC) mechanism is out of range. In this case, the mechanism cannot compensate for the loop delay of the transceiver and successfully compare the delayed received bits to the transmitted ones (see Transceiver delay compensation). The first time that FlexCAN detects the out-of-range condition, this field becomes 1. **NOTE** If CTRL2[BTE] = 1, this field is read as 0 and a write operation has no effect. 0b - In range 1b - Out of range
13 —	Reserved
12-8 TDCOFF	Transceiver Delay Compensation Offset Contains the offset value to be added to the loop delay of the measured transceiver. This value defines the position of the delayed comparison point when bit rate switching is active. See Transceiver delay compensation for details about loop delay measurement. This field can be written in Freeze mode only. Its value can be defined in Protocol Engine Clock periods (CANCLK, see Protocol timing for more details). The value must be smaller than the CAN bit duration in the data bit rate for proper operation. **NOTE** If CTRL2[BTE] = 1, TDCOFF is read as 0 and a write operation has no effect. Do not write 0 to this field.
7-6 —	Reserved
5-0 TDCVAL	Transceiver Delay Compensation Value Contains the value of the transceiver loop delay measured from the transmitted EDL-to-R0 transition edge to the respective received one added to FDCTRL[TDCOFF]. This value is an integer multiple of the Protocol Engine Clock period (CANCLK). If CTRL2[BTE] = 1, this field is read as 0. See Protocol timing for details on the loop delay measurement.

68.6.2.38 CAN FD Bit Timing (FDCBT)

Offset

Register	Offset
FDCBT	C04h

Function

Stores the CAN bit timing variables used in the data phase of CAN FD messages when the FDCTRL[FDRATE] = 1, compatible with the CAN FD specification. Fields in this register define:

- The time quantum duration
- The number of time quanta per CAN bit
- The sample point position for the data bit rate portion of a CAN FD message with BRS = 1

Soft reset does not affect the contents of this register.

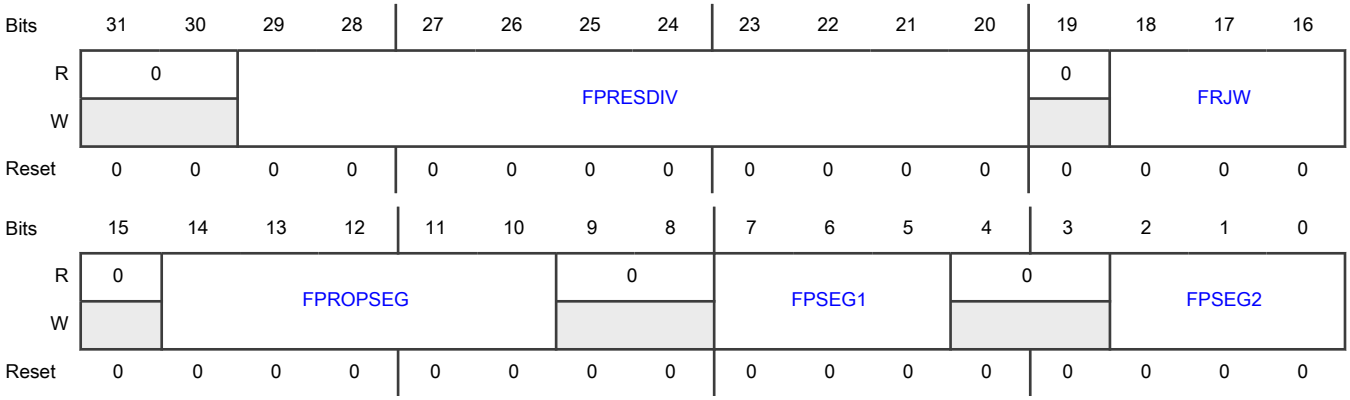
The sum of the Fast Propagation Segment (FPROPSEG) and Fast Phase Segment 1 (FPSEG1) must be at least two time quanta.

Ensure bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1:2015).

NOTE

If CTRL2[BTE] = 1, this register is read as zero and a write operation has no effect.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-20 FPRESDIV	Fast Prescaler Division Factor Defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency in the data bit rate portion of a CAN FD message with BRS = 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. This field can be written only in Freeze mode; the module blocks it in other modes. Sclock frequency = PE clock frequency ÷ (FPRESDIV + 1). **NOTE** To minimize errors when processing FD frames, use the same value for this field and for CTRL1[PRESDIV] or CBT[EPRESDIV]. See the first note in CAN FD frames for details.
19 —	Reserved
18-16 FRJW	Fast Resync Jump Width Defines the maximum number of time quanta that one resynchronization can change a bit time in the data bit rate portion of a CAN FD message with BRS = 1. This field can be written only in Freeze mode; the module blocks it in other modes. Resync Jump Width = FSJW + 1. One Time Quantum = one Sclock period.
15 —	Reserved
14-10 FPROPSEG	Fast Propagation Segment Defines the length of the propagation segment in the bit time in the data bit rate portion of a CAN FD message with BRS = 1. This field can be written only in Freeze mode; the module blocks it in other modes. Propagation Segment Time = FPROPSEG × Time Quanta. One Time Quantum = one Sclock period.
9-8 —	Reserved
7-5 FPSEG1	Fast Phase Segment 1 Defines the length of phase segment 1 in the bit time in the data bit rate portion of a CAN FD message with BRS = 1. This field can be written only in Freeze mode; the module blocks it in other modes. Phase Segment 1 = (FPSEG1 + 1) × Time Quanta. One Time Quantum = one Sclock period.
4-3 —	Reserved
2-0 FPSEG2	Fast Phase Segment 2 Defines the length of phase segment 2 in the data bit rate portion of a CAN FD message with BRS = 1. This field can be written only in Freeze mode; the module blocks it in other modes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Phase Segment 2 = (FPSEG2 + 1) × Time Quanta. One Time Quantum = one Sclock period.

68.6.2.39 CAN FD CRC (FDCRC)

Offset

Register	Offset
FDCRC	C08h

Function

Provides information about the cyclic redundancy check (CRC) of transmitted messages.

FlexCAN uses different CRC polynomials for different frame formats.

- The CRC_15 polynomial is used for all frames in CAN format.
- The CRC_17 polynomial is used for frames in CAN FD format with a DATA FIELD up to 16 bytes.
- The CRC_21 polynomial is used for frames in CAN FD format with a DATA FIELD longer than 16 bytes.

Each polynomial shown below results in a Hamming distance of 6. This register is updated at the same time that the TX Interrupt flag is set.

$$\text{CRC_15} = \text{C599h: } (x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1)$$

$$\text{CRC_17} = \text{3685Bh: } (x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^{6} + x^{4} + x^{3} + x^{1} + 1)$$

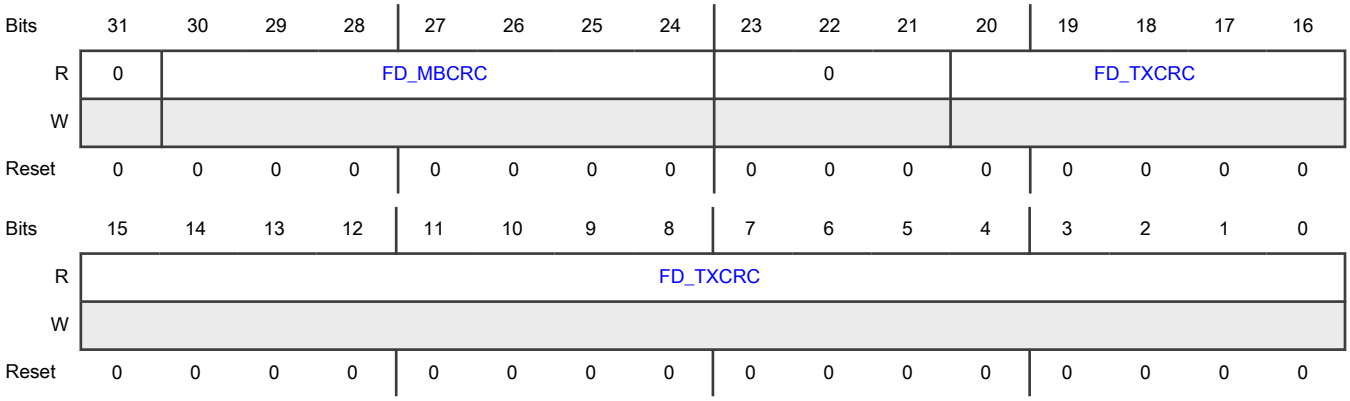
$$\text{CRC_21} = \text{302899h: } (x^{21} + x^{20} + x^{13} + x^{11} + x^{7} + x^{4} + x^{3} + 1)$$

Equation 48. CRC polynomial used on CAN frame

NOTE

See CRC sequence calculation in the CAN Protocol standard (ISO 11898-1:2015) for details.

Diagram



Fields

Field	Function
31 —	Reserved
30-24 FD_MBCRC	CRC Message Buffer Number for FD_TXCRC Indicates the number of the message buffer corresponding to the value in FDCRC[FD_TXCRC], for both FD and non-FD frames. It reports the same information as in CRCR[MBCRC].
23-21 —	Reserved
20-0 FD_TXCRC	Extended Transmitted CRC value Contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. For CRC_15 and CRC_17, the six most significant bits and the four most significant bits are reported as zeroes, respectively. For CRC_15, this field has the same content as Cyclic Redundancy Check (CRCR).

68.6.2.40 Enhanced RX FIFO Control (ERFCR)

Offset

Register	Offset
ERFCR	C0Ch

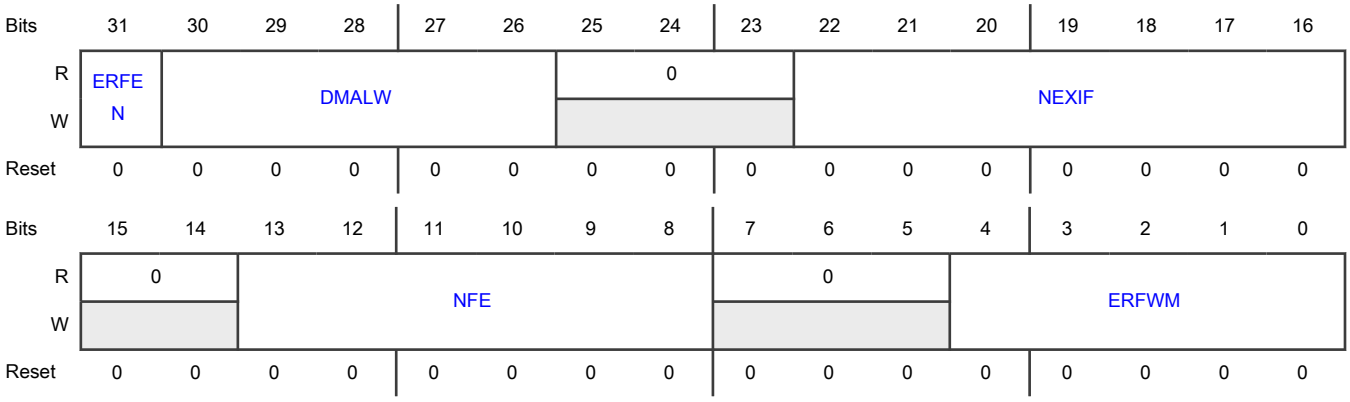
Function

Defines the Enhanced RX FIFO configuration.

This register can be written only in Freeze mode.

Soft reset does not affect any of the contents of this register.

Diagram



Fields

Field	Function
31 ERFEN	Enhanced RX FIFO enable Enables the Enhanced RX FIFO. **NOTE** If MCR[RFEN] = 1, do not write 1 to this field. 0b - Disable 1b - Enable
30-26 DMALW	DMA Last Word Defines the last DMA address for each Enhanced RX FIFO element. This table shows the number of elements and the last address for each Enhanced RX FIFO element according to the value of DMALW.

DMALW	Number of 32-bit words transferred	Last FIFO address
0	1	2000h
1	2	2004h
2	3	2008h
3	4	200Ch
4	5	2010h
5	6	2014h
6	7	2018h
7	8	201Ch
8	9	2020h

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>DMALW</th><th>Number of 32-bit words transferred</th><th>Last FIFO address</th></tr>
<tr><td>9</td><td>10</td><td>2024h</td></tr>
<tr><td>10</td><td>11</td><td>2028h</td></tr>
<tr><td>11</td><td>12</td><td>202Ch</td></tr>
<tr><td>12</td><td>13</td><td>2030h</td></tr>
<tr><td>13</td><td>14</td><td>2034h</td></tr>
<tr><td>14</td><td>15</td><td>2038h</td></tr>
<tr><td>15</td><td>16</td><td>203Ch</td></tr>
<tr><td>16</td><td>17</td><td>2040h</td></tr>
<tr><td>17</td><td>18</td><td>2044h</td></tr>
<tr><td>18</td><td>19</td><td>2048h</td></tr>
</table>
NOTE
Undefined DMALW values in the table are reserved and must not be used.
</td></tr>
<tr><td>25-23
—</td><td>Reserved</td></tr>
<tr><td>22-16
NEXIF</td><td>Number of Extended ID Filter Elements
Defines the number of extended ID filter elements used during the Enhanced RX FIFO matching process.
The value of this field must be less than or equal to NFE + 1.
The number of standard ID filter elements is 2 × (NFE - NEXIF + 1).
This table shows the number of extended ID filters and standard ID filters available for Enhanced RX FIFO if all filter elements are used.
<table>
<tr><th>NEXIF</th><th>NFE</th><th>Number of Extended ID filter elements</th><th>Number of Standard ID filter elements</th></tr>
<tr><td>0</td><td>15</td><td>0</td><td>32</td></tr>
<tr><td>1</td><td>15</td><td>1</td><td>30</td></tr>
<tr><td>2</td><td>15</td><td>2</td><td>28</td></tr>
<tr><td>3</td><td>15</td><td>3</td><td>26</td></tr>
<tr><td>4</td><td>15</td><td>4</td><td>24</td></tr>
<tr><td>5</td><td>15</td><td>5</td><td>22</td></tr>
<tr><td>6</td><td>15</td><td>6</td><td>20</td></tr>
</table>
</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>NEXIF</th><th>NFE</th><th>Number of Extended ID filter elements</th><th>Number of Standard ID filter elements</th></tr>
<tr><td>7</td><td>15</td><td>7</td><td>18</td></tr>
<tr><td>8</td><td>15</td><td>8</td><td>16</td></tr>
<tr><td>9</td><td>15</td><td>9</td><td>14</td></tr>
<tr><td>10</td><td>15</td><td>10</td><td>12</td></tr>
<tr><td>11</td><td>15</td><td>11</td><td>10</td></tr>
<tr><td>12</td><td>15</td><td>12</td><td>8</td></tr>
<tr><td>13</td><td>15</td><td>13</td><td>6</td></tr>
<tr><td>14</td><td>15</td><td>14</td><td>4</td></tr>
<tr><td>15</td><td>15</td><td>15</td><td>2</td></tr>
<tr><td>16</td><td>15</td><td>16</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>15-14
—</td><td>Reserved</td></tr>
<tr><td>13-8
NFE</td><td>Number of Enhanced RX FIFO Filter Elements
Defines the total number of filter elements used during the enhanced RX FIFO matching process according to the table.
<table>
<tr><th>NFE</th><th>Maximum number of extended ID filter elements (NEXIF = NFE + 1)</th><th>Maximum number of standard ID filter elements (NEXIF = 0)</th></tr>
<tr><td>0</td><td>1</td><td>2</td></tr>
<tr><td>1</td><td>2</td><td>4</td></tr>
<tr><td>2</td><td>3</td><td>6</td></tr>
<tr><td>3</td><td>4</td><td>8</td></tr>
<tr><td>4</td><td>5</td><td>10</td></tr>
<tr><td>5</td><td>6</td><td>12</td></tr>
<tr><td>6</td><td>7</td><td>14</td></tr>
<tr><td>7</td><td>8</td><td>16</td></tr>
<tr><td>8</td><td>9</td><td>18</td></tr>
<tr><td>9</td><td>10</td><td>20</td></tr>
<tr><td>10</td><td>11</td><td>22</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>NFE</th><th>Maximum number of extended ID filter elements (NEXIF = NFE + 1)</th><th>Maximum number of standard ID filter elements (NEXIF = 0)</th></tr>
<tr><td>11</td><td>12</td><td>24</td></tr>
<tr><td>12</td><td>13</td><td>26</td></tr>
<tr><td>13</td><td>14</td><td>28</td></tr>
<tr><td>14</td><td>15</td><td>30</td></tr>
<tr><td>15</td><td>16</td><td>32</td></tr>
</table>
</td></tr>
<tr><td>7-5
—</td><td>Reserved</td></tr>
<tr><td>4-0
ERFWM</td><td>Enhanced RX FIFO Watermark
Defines the minimum number of CAN messages stored in the Enhanced RX FIFO. When that number is reached, ERFSR[ERFWMI] becomes 1.
Minimum number of CAN messages = ERFWM + 1.
NOTE
If MCR[DMA] = 1, write 0h to this field.</td></tr>
</table>

68.6.2.41 Enhanced RX FIFO Interrupt Enable (ERFIER)

Offset

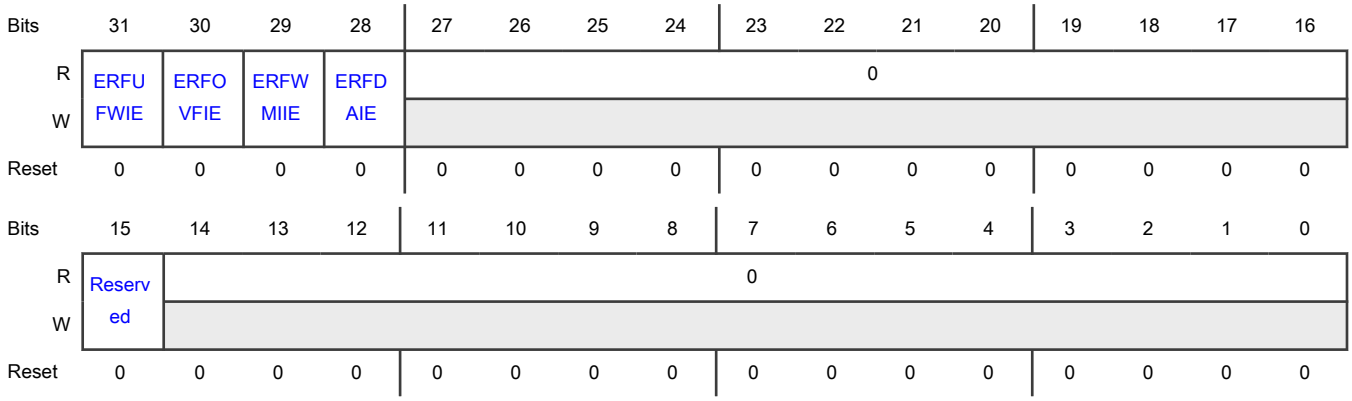
Register	Offset
ERFIER	C10h

Function

Contains the interrupt enables for the Enhanced RX FIFO.

Soft reset does not affect this register.

Diagram



Fields

Field	Function
31 ERFUFWIE	Enhanced RX FIFO Underflow Interrupt Enable Enables interrupt for ERFSR[ERFUFW]. 0b - Disable 1b - Enable
30 ERFOVFIE	Enhanced RX FIFO Overflow Interrupt Enable Enables interrupt for ERFSR[ERFOVF]. 0b - Disable 1b - Enable
29 ERFWMIIE	Enhanced RX FIFO Watermark Indication Interrupt Enable Enables interrupt for ERFSR[ERFWMI]. 0b - Disable 1b - Enable
28 ERFDAIE	Enhanced RX FIFO Data Available Interrupt Enable Enables interrupt for ERFSR[ERFDA]. 0b - Disable 1b - Enable
27-16 —	Reserved
15 —	Reserved
14-0 —	Reserved

68.6.2.42 Enhanced RX FIFO Status (ERFSR)

Offset

Register	Offset
ERFSR	C14h

Function

Contains the status fields of the Enhanced RX FIFO including error indications and a clear FIFO field.

Soft reset does not affect this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERFUFW	ERFOVF	ERFWMI	ERFDA	0	0									ERFE	ERFF
W	W1C	W1C	W1C	W1C	ERFCLR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	0									ERFEL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 ERFUFW	Enhanced RX FIFO Underflow Flag Indicates whether an underflow condition occurred in the enhanced RX FIFO. If ERFIER[ERFUFWIE] = 1, this field generates an interrupt. 0b - No such occurrence 1b - Underflow
30 ERFOVF	Enhanced RX FIFO Overflow Flag Indicates whether an overflow condition occurred in the Enhanced RX FIFO. If ERFIER[ERFOVFIE] = 1, this field generates an interrupt. 0b - No such occurrence 1b - Overflow
29	Enhanced RX FIFO Watermark Indication Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
ERFWMI	Indicates whether the number of messages available in the Enhanced RX FIFO is greater than the watermark defined in ERFCR[ERFWM]. If ERFIER[ERFWMIIE] = 1, this field generates an interrupt. 0b - No such occurrence 1b - Number of messages in FIFO is greater than the watermark
28 ERFDA	Enhanced RX FIFO Data Available Flag Indicates whether there is at least one message stored in the ERX FIFO. If ERFIER[ERFDAIE] = 1, this field generates an interrupt. 0b - No such occurrence 1b - At least one message stored in Enhanced RX FIFO
27 ERFCLR	Enhanced RX FIFO Clear Writing one to this field during Freeze mode clears Enhanced RX FIFO content. Writing to this field outside Freeze mode, or writing 0 to this field, has no effect. 0b - No effect 1b - Clear enhanced RX FIFO content
26-18 —	Reserved
17 ERFE	Enhanced RX FIFO Empty Flag Indicates whether Enhanced RX FIFO is empty. 0b - Not empty 1b - Empty
16 ERFF	Enhanced RX FIFO Full Flag Indicates whether enhanced RX FIFO is full. 0b - Not full 1b - Full
15 —	Reserved
14-6 —	Reserved
5-0 ERFEL	Enhanced RX FIFO Elements Indicates the number of CAN messages stored in the Enhanced RX FIFO.

68.6.2.43 Enhanced RX FIFO Filter Element (ERFFEL0 - ERFFEL31)

Offset

For n = 0 to 31:

Register	Offset
ERFFELn	3000h + (n × 4h)

Function

Stores the filter elements of the Enhanced RX FIFO.

For standard ID filtering, each ERFFEL register stores one filter element. For extended ID filtering, each pair of ERFFEL registers stores one filter element.

ERFFEL registers can be written only in Freeze mode; otherwise, the module blocks them. Reset does not affect these registers. They are located in RAM and must be explicitly initialized prior to any reception.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FEL															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FEL															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 FEL	Filter Element Bits Stores filter elements. Each filter element is used during the match process. If the matching criteria are met, a message is stored in the Enhanced RX FIFO.

68.6.3 Message buffer structure

The message buffer structure used by FlexCAN is represented in the following figure. Both extended (29-bit identifier) and standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual message buffer is 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area 80h–27Fh is used by the message buffers. When CAN FD is enabled, the exact address for each message buffer depends on the size of its payload. See FlexCAN memory partition for CAN FD.

Table 614. Message buffer structure example with 64-byte payload

	31	30	29	28	27	24	23	22	21	20	19	18	17	16	15	8	7	0
0h	EDL	BRS	ESI		CODE			SRR	IDE	RTR	DLC				TIMESTAMP			
4h	PRIO				ID (standard/extended)										ID (extended)			
8h	Data byte 0						Data byte 1								Data byte 2		Data byte 3	
Ch	Data byte 4						Data byte 5								Data byte 6		Data byte 7	
10h	Data byte 8						Data byte 9								Data byte 10		Data byte 11	
14h	Data byte 12						Data byte 13								Data byte 14		Data byte 15	
18h	Data byte 16						Data byte 17								Data byte 18		Data byte 19	
1Ch	Data byte 20						Data byte 21								Data byte 22		Data byte 23	
20h	Data byte 24						Data byte 25								Data byte 26		Data byte 27	
24h	Data byte 28						Data byte 29								Data byte 30		Data byte 31	
28h	Data byte 32						Data byte 33								Data byte 34		Data byte 35	
2Ch	Data byte 36						Data byte 37								Data byte 38		Data byte 39	
30h	Data byte 40						Data byte 41								Data byte 42		Data byte 43	
34h	Data byte 44						Data byte 45								Data byte 46		Data byte 47	
38h	Data byte 48						Data byte 49								Data byte 50		Data byte 51	
3Ch	Data byte 52						Data byte 53								Data byte 54		Data byte 55	
40h	Data byte 56						Data byte 57								Data byte 58		Data byte 59	
44h	Data byte 60						Data byte 61								Data byte 62		Data byte 63	

= Unimplemented or reserved

Table 615. Field descriptions

Mnemonic	Field	Description
EDL	Extended Data Length	Distinguishes between CAN format and CAN FD format frames. EDL must not be 1 for message buffers configured to RANSWER with code field 1010b (see Table 616).
BRS	Bit Rate Switch	Defines whether the bit rate is switched inside a CAN FD format frame.
ESI	Error State Indicator	Indicates whether the transmitting node is error-active or error-passive.
CODE	Message Buffer Code	Can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 616 and Table 617. See Functional description.

Table 616. Message buffer code for RX buffers

CODE description	RX code BEFORE RX new frame	SRV[1]	RX code AFTER successful reception[2]	RRS[3]	Comment
0000b: INACTIVE. Message buffer is not active.	INACTIVE	—	—	—	Message buffer does not participate in the matching process.
0100b: EMPTY. Message buffer is active and empty.	EMPTY	—	FULL	—	When a frame is received successfully (after Move-in), CODE is automatically updated to FULL.
0010b: FULL. Message buffer is full.	FULL	Yes	FULL	—	The act of reading the Control and Status word followed by unlocking the message buffer (SRV) does not make CODE return to EMPTY. It remains FULL. If a new frame is moved to the message buffer after the message buffer is serviced, the code remains FULL. See Matching process for matching details related to FULL code.
		No	OVERRUN	—	If the message buffer is FULL and a new frame is moved to this message buffer before the CPU services it, CODE is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0110b: OVERRUN. Message buffer is being overwritten into a full buffer.	OVERRUN	Yes	FULL	—	If CODE indicates OVERRUN and the CPU has serviced the message buffer, when a new frame is moved to the message buffer, CODE returns to FULL.
		No	OVERRUN	—	If CODE already indicates OVERRUN, and another new frame must be moved, the message buffer is overwritten

Table continues on the next page...

Table 616. Message buffer code for RX buffers (continued)

CODE description	RX code BEFORE RX new frame	SRV[1]	RX code AFTER successful reception[2]	RRS[3]	Comment
					again, and CODE remains OVERRUN. See Matching process for details about overrun behavior.
1010b: RANSWER[4]. A frame was configured to recognize a Remote Request frame and transmit a Response frame in return.[5]	RANSWER	—	TANSWER (1110b)	0	A Remote Answer was configured to recognize a Remote Request frame received. After that, a message buffer is set to transmit a response frame. CODE is automatically changed to TANSWER (1110b). See Matching process for details. If CTRL2[RRS] = 0, transmit a response frame when a remote request frame with the same ID is received.
		—	—	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0] = 1: BUSY. FlexCAN is updating the contents of the message buffer. The CPU must not access the message buffer.	BUSY[6]	—	FULL	—	Indicates that the message buffer is being updated. It automatically becomes 0 and does not interfere with the next CODE.
		—	OVERRUN	—	

1. SRV: Serviced message buffer. Message buffer was read and unlocked by reading TIMER or other message buffer.
2. A frame is considered a successful reception after the frame is moved to a message buffer (move-in process). See Move-in.
3. Remote Request Stored field. See Control 2 (CTRL2).
4. Code 1010b is not considered TX and a message buffer with this code should not be aborted.
5. Code 1010b must be used in message buffers configured in CAN FD format, with EDL = 1.
6. For TX message buffers, the BUSY bit should be ignored upon read, except when MCR[AEN] = 1. If this field is 1, the corresponding message buffer does not participate in the matching process.

Table 617. Message buffer code for TX buffers

CODE Description	TX Code BEFORE TX frame	MB RTR	TX Code AFTER successful transmission	Comment
1000b: INACTIVE. Message buffer is not active.	INACTIVE	—	—	Message buffer does not participate in arbitration process.
1001b: ABORT. Message buffer is aborted.	ABORT	—	—	Message buffer does not participate in arbitration process.
1100b: DATA. Message buffer is a TX data frame (MB RTR must be 0).	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the message buffer automatically returns to the INACTIVE state.
1100b: REMOTE. Message buffer is a Transmit Remote Request frame (MB RTR must be 1).	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the message buffer automatically becomes an RX Empty message buffer with the same ID.
1110b: TANSWER. Message buffer is a Transmit Response frame from an incoming Remote Request frame.	TANSWER	—	RANSWER	This intermediate code is automatically written to the message buffer by the CHI as a result of a match to a Remote Request frame. The Remote Response frame is transmitted unconditionally once, then the code automatically returns to RANSWER (1010b). The CPU can also write this code with the same effect. The Remote Response frame can be a data frame or another remote request frame, depending on the value of RTR. See Matching process and Arbitration process for details.

Table 618. RX and TX message buffer field descriptions

Mnemonic	Field	Description
SRR	Substitute Remote Request	Fixed recessive bit, used only in extended format. Write 1 to SRR for transmission (TX Buffers). SRR is stored with the value received on the CAN

Table continues on the next page...

Table 618. RX and TX message buffer field descriptions (continued)

Mnemonic	Field	Description
		bus for RX receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, it is interpreted as an arbitration loss. 1: Recessive value is compulsory for transmission in extended format frames. 0: Dominant is not a valid value for transmission in extended format frames.
IDE	ID Extended Bit	Identifies whether the frame format is standard or extended. 1: Frame format is extended 0: Frame format is standard
RTR	Remote Transmission Request	Affects the behavior of remote frames and is part of the reception filter. See Table 616, Table 617, and CTRL2[RRS]. If FlexCAN transmits this field as 1 (recessive) and receives it as 0 (dominant), it is interpreted as an arbitration loss. If this field is transmitted as 0 (dominant) and it is received as 1 (recessive), FlexCAN treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission. 1: If message buffer is TX, indicates that the current message buffer may have a Remote Request frame to be transmitted. If the message buffer is RX, incoming remote request frames may be stored. 0: Indicates that the current message buffer has a Data frame to be transmitted. In an RX message buffer, it may be considered in matching processes. **NOTE** When configuring CAN FD frames, this field must be 0.
DLC	Data Length Code	Indicates the length (in bytes) of the RX or TX data, which is located in offset 8h–Fh of the message buffer space (see Table 614). In reception, this field is written by FlexCAN, copied from the DLC field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 619).
TIMESTAMP	Free-Running Counter Timestamp	Provides a copy of the Free-Running Timer, captured for TX and RX frames when the beginning of the Identifier field appears on the CAN bus.
PRIO	Local priority	Used only when MCR[LPRIOEN] = 1, and only makes sense for transmit message buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.
ID	Frame Identifier	In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

Table continues on the next page...

Table 618. RX and TX message buffer field descriptions (continued)

Mnemonic	Field	Description
DATA BYTE 0–63	Data Field	Up to 64 bytes can be used for a data frame, depending on the size of payload selected for the message buffers. For RX frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC, as shown in Table 619.

Table 619. DATA BYTE validity

DLC	Valid data bytes
0	None
1	DATA BYTE 0
2	DATA BYTE 0–1
3	DATA BYTE 0–2
4	DATA BYTE 0–3
5	DATA BYTE 0–4
6	DATA BYTE 0–5
7	DATA BYTE 0–6
8	DATA BYTE 0–7
9	DATA BYTE 0–11
10	DATA BYTE 0–15
11	DATA BYTE 0–19
12	DATA BYTE 0–23
13	DATA BYTE 0–31
14	DATA BYTE 0–47
15	DATA BYTE 0–63

68.6.4 FlexCAN memory partition for CAN FD

When CAN FD is enabled, FlexCAN RAM can be partitioned into blocks of 512 bytes each. Each block can accommodate a number of message buffers depending on the configuration provided by FDCTRL[MBDSR*n*] as shown in Table 620.

Table 620. RAM partition

RAM block	Number of MBs with 8 bytes (default range)	Size control field in FDCTRL	Number of MBs of different sizes, per block
0	0 to 31	MBDSR0	MBDSR0 = 00, 32 MBs with 8-byte payload MBDSR0 = 01, 21 MBs with 16-byte payload MBDSR0 = 10, 12 MBs with 32-byte payload MBDSR0 = 11, 7 MBs with 64-byte payload

Payload sizes of 16, 32, or 64 bytes may be configured in some or all of RAM blocks. In those cases, the total number of MBs and their respective number order may differ from the default configuration of 8 bytes. Consider an example where:

- Block0 is configured to an 8-byte payload
- Block1 is configured to a 16-byte payload

In this case, Table 621 indicates how the message buffers are arranged in RAM.

Table 621. RAM partition example

RAM block	Payload size	Number of MBs in the RAM block	Message buffer range
0	FDCTRL[MBDSR0] = 00, 8-byte payload	32	0 to 31

68.6.5 FlexCAN message buffer memory map

The FlexCAN memory buffers are allocated in memory according to the tables below.

Table 622. 8-byte message buffers

Address offset (hex)	MBDSR = b00 8-byte payload
0080	MB0
0090	MB1
00A0	MB2
00B0	MB3
00C0	MB4
00D0	MB5
00E0	MB6
00F0	MB7
0100	MB8
0110	MB9
0120	MB10
0130	MB11
0140	MB12
0150	MB13
0160	MB14
0170	MB15
0180	MB16
0190	MB17
01A0	MB18

Table continues on the next page...

Table 622. 8-byte message buffers (continued)

Address offset (hex)	MBDSR = b00 8-byte payload
01B0	MB19
01C0	MB20
01D0	MB21
01E0	MB22
01F0	MB23
0200	MB24
0210	MB25
0220	MB26
0230	MB27
0240	MB28
0250	MB29
0260	MB30
0270	MB31

Table 623. 16-byte message buffers

Address offset (hex)	MBDSR = b01 16-byte payload
0080	MB0
0098	MB1
00B0	MB2
00C8	MB3
00E0	MB4
00F8	MB5
0110	MB6
0128	MB7
0140	MB8
0158	MB9
0170	MB10
0188	MB11
01A0	MB12
01B8	MB13

Table continues on the next page...

Table 623. 16-byte message buffers (continued)

Address offset (hex)	MBDSR = b01 16-byte payload
01D0	MB14
01E8	MB15
0200	MB16
0218	MB17
0230	MB18
0248	MB19
0260	MB20

Table 624. 32-byte message buffers

Address offset (hex)	MBDSR = b10 32-byte payload
0080	MB0
00A8	MB1
00D0	MB2
00F8	MB3
0120	MB4
0148	MB5
0170	MB6
0198	MB7
01C0	MB8
01E8	MB9
0210	MB10
0238	MB11

Table 625. 64-byte message buffers

Address offset (hex)	MBDSR = b11 64-byte payload
0080	MB0
00C8	MB1
0110	MB2
0158	MB3

Table continues on the next page...

Table 625. 64-byte message buffers (continued)

Address offset (hex)	MBDSR = b11 64-byte payload
01A0	MB4
01E8	MB5
0230	MB6

68.6.6 Legacy RX FIFO structure

When MCR[RFEN] = 1, the memory area 80h–DCh (which is normally occupied by MBs 0–5) is used by the reception Legacy FIFO engine.

The region 80h–8Ch contains the output of the Legacy RX FIFO, which the CPU must read as a message buffer. This output contains the oldest message that has been received but not yet read. The region 90h–DCh is reserved for internal use of the Legacy RX FIFO engine.

An additional memory area, which starts at E0h and may extend up to 2DCh (normally occupied by MBs 6–37) depending on the value of CTRL2[RFFN], contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the Legacy RX FIFO.

Out of reset, the ID filter table flexible memory area defaults to E0h and extends only to FCh, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Legacy RX FIFO data structure.

Table 626. Legacy RX FIFO structure

	31 28 24	23	22	21	20	19 18 17 16	15 8	7 0
80h	IDHIT		SRR	IDE	RTR	DLC	TIMESTAMP	
84h		ID standard				ID extended		
88h	Data byte 0		Data byte 1				Data byte 2	Data byte 3
8Ch	Data byte 4		Data byte 5				Data byte 6	Data byte 7
90h–DCh	Reserved							
E0h	ID filter table element 0							
E4h	ID filter table element 1							
E8h–2D4h	ID filter table elements 2 to 125							
2D8h	ID filter table element 126							
2DCh	ID filter table element 127							
	= Unimplemented or reserved							

Each ID filter table element occupies an entire 32-bit word. One, two, or four Identifier Acceptance Filters (IDAF) can compound each element, depending on MCR[IDAM]. The following tables show the IDAF indexing.

Table 627 shows the three different formats of the ID table elements. All elements of the table must have the same format. See Legacy RX FIFO for more information.

Table 627. ID table structure

Format	31	30	29 24	23 16	15	14	13 8	7 1	0
A	RTR	IDE	RXIDA (standard = 29–19, extended = 29–1)						
B	RTR	IDE	RXIDB_0 (standard = 29–19, extended = 296–)		RTR	IDE	RXIDB_1 (standard = 13–3, extended = 13–0)		
C	RXIDC_0 (std and ext = 31–24)		RXIDC_1 (std and ext = 23–16)		RXIDC_2 (std and ext = 15–8)			RXIDC_3 (std and ext = 7–0)	
		= Unimplemented or Reserved							

Table 628. Field descriptions

Mnemonic	Field	Description
RTR	Remote Frame	Specifies whether remote frames are accepted into the Legacy FIFO if they match the target ID. 1: Remote frames can be accepted and data frames are rejected. 0: Remote frames are rejected and data frames can be accepted.
IDE	Extended Frame	Specifies whether extended or standard frames are accepted into the Legacy FIFO if they match the target ID. 1: Extended frames can be accepted and standard frames are rejected. 0: Extended frames are rejected and standard frames can be accepted.
RXIDA	RX Frame Identifier (Format A)	Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.
RXIDB_0, RXIDB_1	RX Frame Identifier (Format B)	Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.
RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3	RX Frame Identifier (Format C)	Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.
IDHIT	Identifier Acceptance Filter Hit Indicator	Indicates which identifier acceptance filter the received message in the output of the Legacy RX FIFO hit. See Legacy RX FIFO for more information.

68.6.7 Enhanced RX FIFO structure

When ERFCR[ERFEN] = 1, the Enhanced RX FIFO is enabled. The region 2000h–2048h contains the output of the Enhanced RX FIFO, which the CPU must read as a message buffer. This output contains the oldest message that has been received but not yet read.

Table 629. Enhanced RX FIFO structure

	31	30	29	28	24	23	22	21	20	19	18	17	16	15	8	7	6	0
2000h	EDL	BRS	ESI	Reserved			SRR	IDE	RTR	DLC				TIMESTAMP LEGACY				
2004h	Reserved			ID (standard/extended)										ID (extended)				
2008h	Data byte 0						Data byte 1							Data byte 2			Data byte 3	
200Ch	Data byte 4						Data byte 5							Data byte 6			Data byte 7	
2010h	Data byte 8						Data byte 9							Data byte 10			Data byte 11	
2014h	Data byte 12						Data byte 13							Data byte 14			Data byte 15	
2018h	Data byte 16						Data byte 17							Data byte 18			Data byte 19	
201Ch	Data byte 20						Data byte 21							Data byte 22			Data byte 23	
2020h	Data byte 24						Data byte 25							Data byte 26			Data byte 27	
2024h	Data byte 28						Data byte 29							Data byte 30			Data byte 31	
2028h	Data byte 32						Data byte 33							Data byte 34			Data byte 35	
202Ch	Data byte 36						Data byte 37							Data byte 38			Data byte 39	
2030h	Data byte 40						Data byte 41							Data byte 42			Data byte 43	
2034h	Data byte 44						Data byte 45							Data byte 46			Data byte 47	
2038h	Data byte 48						Data byte 49							Data byte 50			Data byte 51	
203Ch	Data byte 52						Data byte 53							Data byte 54			Data byte 55	
2040h	Data byte 56						Data byte 57							Data byte 58			Data byte 59	
2044h	Data byte 60						Data byte 61							Data byte 62			Data byte 63	
IH_OFF	Reserved																	ID HIT
204Ch	11 Enhanced FIFO Elements (Reserved)																	
...																		
238Ch																		

NOTE

ID HIT offset change dynamically according to data length code (DLC) as shown in Table 630.

Table 630. ID HIT offset

Data Length Code (DLC)	ID HIT offset (IH_OFF)
0	2008h
1–4	200Ch

Table continues on the next page...

Table 630. ID HIT offset (continued)

Data Length Code (DLC)	ID HIT offset (IH_OFF)
5–8	2010h
9	2014h
10	2018h
11	201Ch
12	2020h
13	2028h
14	2038h
15	2048h

Table 631. Field descriptions

Mnemonic	Field	Description
EDL	Extended Data Length	Distinguishes between classical CAN format and CAN FD format frames. 0: Classical CAN frame format 1: CAN FD frame format
BRS	Bit Rate Switch	Defines whether the bit rate is switched inside a CAN FD format frame. 0: Bit rate is not switched in a CAN FD frame. 1: Bit rate is switched in a CAN FD frame.
ESI	Error State Indicator	Indicates whether the transmitting node is error-active or error-passive. This field is meaningful only if EDL = 1. 0: Error-active 1: Error-passive
SRR	Substitute Remote Request	Fixed recessive bit, used only in extended format. Transmitting nodes always send it as recessive and receiving nodes can receive it as either recessive or dominant. If FlexCAN receives this bit as dominant, it is interpreted as an arbitration loss.
IDE	ID Extended Bit	Identifies whether the frame format is standard or extended. 0: Standard 1: Extended
RTR	Remote Frame	Identifies whether the current frame is a data frame or a remote request. 0: Data frame 1: Remote request
DLC	Data Length Code	Defines the number of bytes in the data field of a CAN frame (Data byte 0 to Data byte 63). When RTR = 1, the frame is a remote request and does

Table continues on the next page...

Table 631. Field descriptions (continued)

Mnemonic	Field	Description
		not include the data field, regardless of the DLC field. See Table 619 for more details.
TIMESTAMP	16-bit Timestamp	Provides a copy of the Free-Running Timer, captured during the CAN frame.
ID	Frame Identifier	In base frame format, only the 11 most significant bits are used for frame identification. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification.
DATA BYTE 0–63	Data Field	Up to 64 bytes can be stored in the data field.
IDHIT	Identifier Acceptance Filter Hit Indicator	Indicates which Enhanced RX FIFO Filter Element (ERFFEL0 - ERFFEL31) the received message in the output of the Enhanced RX FIFO hit. For each filter region, standard-ID filter space, and extended-ID filter space, there is an independent index starting from zero. Table 632 shows how FlexCAN writes IDHIT according to each filter element.

Table 632. IDHIT for Enhanced RX FIFO

Enhanced RX FIFO filter element - ERFFEL	IDHIT value	Filter element type
ERFFEL0	0	Extended-ID
ERFFEL1	1	Extended-ID
. . .	. . .	Extended-ID
ERFFEL(m-1)	m-1	Extended-ID
ERFFEL(m)	0	Standard-ID
ERFFEL(m+1)	1	Standard-ID
. . .	. . .	Standard-ID
ERFFEL(2n-m+1)	2x(n-m)+1	Standard-ID

NOTE

Where m = NEXIF and n = NFE. If NEXIF = 0, only standard-ID filter elements exist. If NEXIF > NFE, only extended-ID filter elements exist.

Chapter 69
EMVSIM

69.1 Chip-specific EMVSIM information

Table 633. Reference links to related information

Topic	Related module	Reference
Full description	EMVSIM	EMVSIM
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

69.1.1 Module instances

This device contains two instances of the EMVSIM module, EMVSIM0 and EMVSIM1.

69.1.2 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

69.2 Overview

EMVSIM facilitates communication with smart cards that are compatible with these standards:

- EMV v4.3 (Book 1)
- ISO/IEC 7816-3

69.2.1 Block diagram

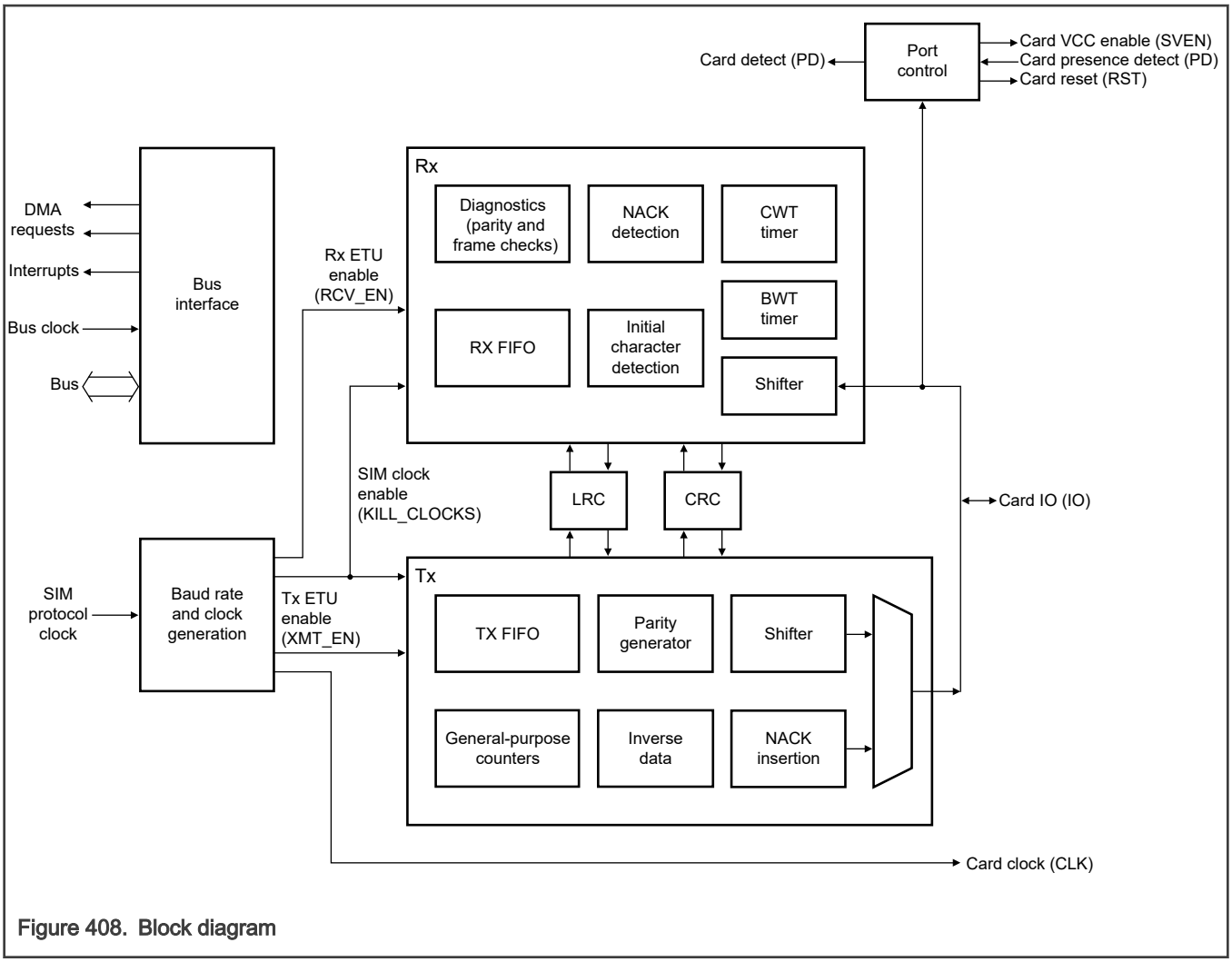


Figure 408. Block diagram

69.2.2 Features

- Independent clock for SIM logic (transmitter and receiver)
- Independent clock for register read-write interface
- 16-byte–deep FIFO for transmitter and receiver
- Automatic NACK generation on parity error and receiver FIFO overflow error
- Support for both inverse and direct conventions
- Retransmission of a byte on smart card NACK request with programmable threshold of re-transmissions
- Auto-detection of initial character in receiver, and setting of data format (inverse or direct)
- NACK detection in receiver
- Independent timers to measure:
  - Character wait time
  - Block wait time
  - Block guard time

- Two general-purpose counters available for use by software, with programmable clock selection for the counters
- DMA support for transferring data to and from FIFOs, with a programmable option to select interrupt or DMA
- Programmable prescaler generating the selected frequency for card clock and baud rate divisor, generating internal ETU clocks for a transmitter and receiver for any F/D ratio
- Manual control of all smart card interface signals
- Automatic power-down of port logic on smart card presence detection
- 8-bit LRC and 16-bit CRC generation for bytes sent out from transmitter, and checking incoming message checksum for receiver

69.3 Functional description

69.3.1 Bus interface

EMVSIM is a block that is configured via the chip's peripheral bus interface. EMVSIM has 32-bit–aligned memory-mapped registers that are used to configure and control its operations. You can write the transmit bytes or read the received bytes using either interrupt operation or DMA operation. EMVSIM can generate necessary DMA requests for the operation.

69.3.2 Smart card interface and control

EMVSIM controls all interface signals for the smart card. The smart card clock is generated by using the prescaler (CLKCFG[CLK_PRSC]) voltage enable through Port Control and Status (PCSR). The message bytes are transmitted and received on the card I/O device pin (IO) which is used in open drain configuration.

The smart card interface signals, except IO (controlled by the transmitter and receiver), are generated by the port control block. The port control block controls the assertion and de-assertion of the clock, reset, and SVEN signals to the smart card. It can do so manually (software explicitly writes the necessary bits) or by using auto power-down. In auto power-down, the module can itself power down the smart card when you configure it to do so by software. The port control block can also detect the presence of the smart card and generate the necessary interrupts to indicate its presence.

For more information about powering up and automatic power down, see Powering up and Automatic power-down sections.

69.3.2.1 Smart card presence detect

The smart card presence detect pin PD (see External signals) allows detecting the insertion or removal of a smart card. PCSR[SPDES] allows you to configure the software as to which edge of the PD pin causes a presence detect interrupt flag to be asserted.

When the smart card is removed, an internal pull-down is required on the PD pin to allow a high-to-low transition on this pin.

69.3.3 EMVSIM receiver

The EMVSIM receiver is designed to receive all message bytes and store valid bytes into the receive FIFO. It performs checks on incoming messages and tells the transmitter to enable NACK insertion for each message. In addition, the receiver automatically detects NACKs received when the transmitter is sending out message bytes.

The receiver includes a 16-byte deep FIFO to store the correctly received bytes from the smart card. You must configure software to ensure that it reads out the data from the FIFO, preventing any overflow when more bytes are sent from the card.

You can configure the receiver to detect the initial character. When configured to do so, the data format (direct or inverse convention) is also determined by the receiver. Then EMVSIM sets the appropriate field in the register map.

Internal timers in the receiver measure the various wait times (character, block, and guard). After receiving a byte, the receiver checks the byte for a parity error (in T = 0 mode) and framing error. It can then direct the transmitter to insert NACK on parity error detection. For a block of bytes received, the receiver checks for LRC or CRC errors (in T = 1 mode). See LRC and CRC for more LRC and CRC information. Other errors are stored as status for software to be aware of their occurrence. The status of the receiver operation is indicated to software via maskable interrupts.

69.3.3.1 Data sampling

The incoming data is internally synchronized and sampled by using a 16x oversampled clock, called the EMVSIM receiver logic clock. The frequency of this clock is 16 times the ETU clock (1/ETU period). The whole receiver works on this EMVSIM receiver logic clock.

69.3.3.2 Initial character detection

EMVSIM supports detection of the initial character and determines the data format when it is placed in the initial character mode by writing 1 to CTRL[ICM]. When placed in this mode, EMVSIM samples the input line and waits until it has detected one of the initial characters. After detecting one, EMVSIM writes 0 to CTRL[ICM]. It also writes the appropriate value to CTRL[IC], depending on the data format detected using the initial character.

The two possible data formats are inverse convention (Figure 409) and direct convention (Figure 410). Essentially, inverse convention differs from direct convention in that the order of the bits is flipped (in other words, MSB for LSB) and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

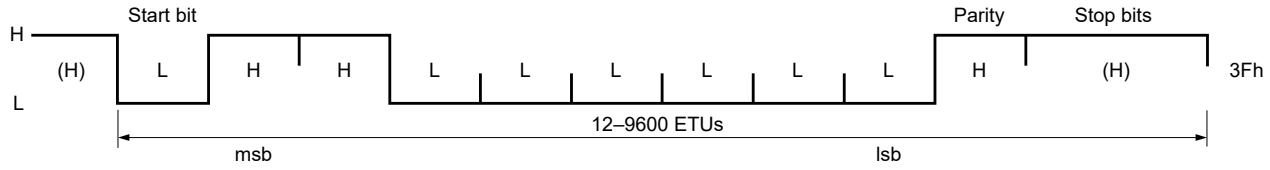


Figure 409. Initial character in inverse convention

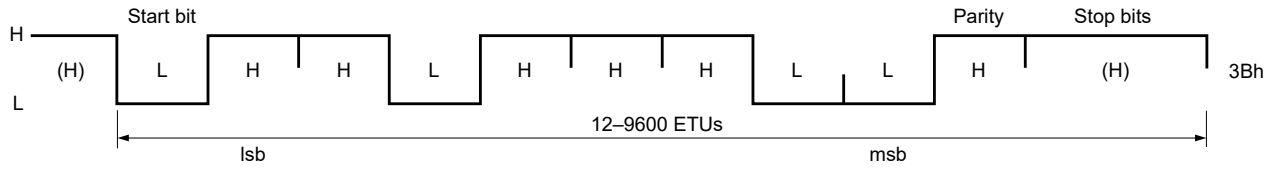


Figure 410. Initial character in direct convention

To place EMVSIM into initial character mode, write 1 to CTRL[ICM]. After a valid initial character is received, EMVSIM accordingly writes 1 to CTRL[IC] and writes 0 to CTRL[ICM]. You can read CTRL[IC] to determine which mode EMVSIM is currently using. The 3Bh (as decoded by direct convention) with parity bit high causes direct convention to be used (CTRL[IC] = 0), whereas a 3Fh (as decoded by inverse convention) with parity bit high causes inverse convention to be used (CTRL[IC] = 1).

The receiver remains in initial character mode until you disable it or until a valid initial character is detected. Upon detection of a valid character, the receiver enters data receive mode. NXP recommends that you use one of the general-purpose counters as a timeout during initial character mode.

When the receiver is in initial character mode, all bytes received continue to be placed into the receive FIFO whether they are valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes are decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if CTRL[IC] is low, and the correct initial character for setting inverse convention is received, that character and all subsequently received characters are stored in the FIFO after having been decoded using inverse convention (for example, the initial character is stored as 3Fh).

NOTE

There is a condition where a parity error in the initial byte for direct convention (3Bh) might be decoded as what appears to be a valid initial character for inverse convention (that is, 3Fh). EMVSIM does not recognize this as a valid initial character for inverse convention and writes 1 to RX_STATUS[PEF].

69.3.3.3 Receiver diagnostics

EMVSIM performs diagnostic checks on the incoming messages. These include:

- Parity checks
- Message frame checks

Parity error detection

The receiver is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented, making the data appear to be odd parity. The parity bit is defined as the tenth bit of the received data. The parity of the second through tenth received bits is calculated by the receiver parity logic. This logic determines if the parity of the nine received bits is correct.

When EMVSIM detects a parity error on a given byte, the parity error flag (RX_STATUS[PEF]) is asserted if NACK generation on errors is disabled (CTRL[ANACK] = 0). The parity error flag does not generate an interrupt to the system. However, EMVSIM can cause the transmitter to transmit a NACK (if CTRL[ANACK] = 1) and request retransmission of the current byte. Software does not need to enable the transmitter to perform this action.

Framing error detection

The receiver is responsible for detecting framing errors in the received data. Data is always transmitted with 2 STOP bits, except in 11 ETU mode, where there is one STOP bit. The STOP bits are defined as high state at the 11th and 12th bit positions in the received data. In either mode, the receiver expects to see a high at the 11th bit position in the received data. If a low is detected at the 11th bit position, the receiver indicates a framing error for that byte.

When a framing error is detected on a given byte, EMVSIM writes 1 to RX_STATUS[FEF]. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the smart card asking for a retransmission of the corrupted data.

LRC and CRC

The receiver also checks the LRC or CRC on the incoming block of bytes (if LRC or CRC is enabled). The result of the CRC or LRC check is updated in the RX_STATUS register after the complete message block is received. See LRC and CRC for more LRC and CRC information.

69.3.3.4 NACK detection

During transmission of message bytes, the receiver automatically checks for NACKs sent by the smart card. The software does not need to enable the receiver for this purpose.

The NACK is detected by sampling the IO line at 11 ETUs after the falling edge of the start bit. After a NACK is detected, the transmitter is signaled. The transmitter then retransmits the current byte after two ETU intervals (as per ISO 7816 specification).

69.3.3.5 Using EMVSIM receiver with T = 1 smart cards

EMVSIM provides hardware support for T = 1 type smart cards. This type of card presents several requirements above and beyond the standard T = 0 cards. The features provided to meet the requirements that pertain to the EMVSIM receiver are:

Table 634. Receiver features and requirements

Feature	Description
11 ETU characters	T = 1 cards can transmit with character lengths of 11 ETUs (that is, one STOP bit). EMVSIM provides CTRL[RCVR_11] to configure the receiver state machine to accept 11 ETU characters.
Character wait time counter	The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters received from the smart card. The value of CWT is from 12–32779 ETU. To identify when the smart card has exceeded CWT, EMVSIM provides a 16-bit counter with programmable comparator clocked at the ETU bit rate.

Table continues on the next page...

Table 634. Receiver features and requirements (continued)

Feature	Description
Block wait time	The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The block wait time must not exceed a value that you program. EMVSIM provides a 32-bit block wait timer that you can use to identify when the BWT has been exceeded.
Block guard time	The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The block guard must be greater than a value that you program. EMVSIM provides a 16-bit block guard timer that you can use to identify when the BGT has been exceeded.
Error detection code	T = 1 cards can specify LRC or CRC error detection codes to use. EMVSIM provides hardware support for both the LRC and CRC operations.

69.3.4 EMVSIM transmitter

The transmitter is responsible for:

- Reading the message bytes from the transmit FIFO
- Adding the LRC or CRC at the end of the message block
- Retransmitting any byte on reception of a NACK

It assists the receive operation by inserting NACKs when directed by the receiver.

The transmitter includes a 16-byte deep FIFO to store the bytes sent to the smart card. These bytes are sent to the smart card serially by the transmitter. The selected data format (inverse or direct conversion) is supported, and EMVSIM converts the byte into the correct data format before it is shifted out serially.

The transmitter also supports retransmission of the current byte on the reception of a NACK. The transmitter automatically inserts a NACK when the receiver indicates a parity or FIFO overflow error, if you have configured software to do so. The transmitter is not required to be enabled for this.

The transmitter also includes a timer to insert the programmable guard time between the transmitter bytes, and general purpose timers to allow software to perform certain measurements.

The status of the transmitter operation is provided to software via maskable interrupts.

69.3.4.1 Message transmission

The transmitter operation does not start until the transmit FIFO is empty. Software can configure EMVSIM and enable the transmitter, but no transmission starts until a byte has been written into the transmit FIFO. Writing zero to CTRL[XMT_EN] while the transmitter is in operation halts any transmission in progress, and flushes the transmit FIFO. See Transmit FIFO for more details.

The transmitter continuously transmits the messages from the transmit FIFO. Upon NACK detection, the transmitter resends the same byte until the transmitter NACK threshold limit is reached. When TX_STATUS[TNTE] = 1, then:

- EMVSIM halts the transmitter.
- All pending transfers are aborted.
- EMVSIM writes 1 to:
  - TX_STATUS[TCF]
  - TX_STATUS[ETCF]
  - TX_STATUS[TFE]
- All bytes remaining in the transmit FIFO are lost.

There is no way to restart the transmission on the next byte in the FIFO. The transmitter remains frozen until software writes 1 to TX_STATUS[TNTE]. The only way to clear TX_STATUS[TNTE] is to write a 1 to it.

It is possible to disable the detection of NACKs from the smart card by writing 0h to TX_THD[TNCK_THD]. By writing 1h to TNCK_THD, it is possible to disable all retransmissions while still setting TX_STATUS[TNTE] on the first NACK received. In general, TX_STATUS[TNTE] becomes 1 on the NACK that causes the threshold controlled by TX_THD[TNCK_THD] to be reached. This final NACK does not cause a retransmission, whereas all previous NACKs cause a retransmission.

If all bytes are successfully transmitted and the transmit FIFO is empty, then the transmitter sends the check byte (that is, LRC or CRC) if the XMT_CRC_LRC and CTRL[LRC_EN] or CTRL[CRC_EN] = 1. The transmitter sends each byte to the LRC or CRC block to allow check byte generation.

At the end of each byte, the transmitter inserts the programmed guard ETUs. See Transmit guard time for details.

69.3.4.2 NACK insertion

The transmitter is responsible for driving the output low during the STOP bit time to signify that an error was detected in the received data from the smart card. This logic responds to a NACK request generated by the receiver block.

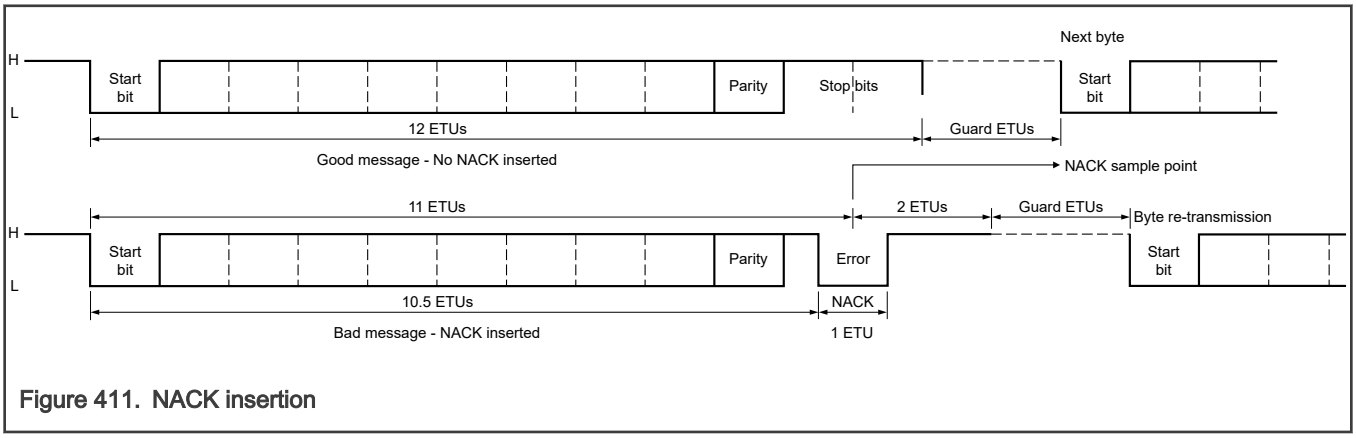


Figure 411. NACK insertion

The NACK generation logic is also responsible for keeping track of the number of NACKs received during a transmit operation. The receiver detects NACKs generated by the smart card, and then reports them to the transmit NACK logic. After the number of detected NACKs has reached the programmed threshold (TX_THD[TNCK_THD]), then:

- An interrupt flag (TX_STATUS[TNTE]) is generated.
- The transmit FIFO is flushed.
- The transmitter is disabled.

69.3.4.3 Using the EMVSIM transmitter with T = 1 smart cards

EMVSIM provides hardware support for T = 1 type smart cards. These type of cards present several requirements above and beyond the standard T = 0 cards. The features provided to meet the requirements that pertain to the EMVSIM transmitter are:

Table 635. Transmitter features and requirements

Feature	Description
11 ETU characters	The EMVSIM transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 (FFh) in Transmitter Guard ETU Value (TX_GETU) sets the number of ETUs per character transmitted to 11.
Character waiting time	CWT is defined as the time between the start bits of two consecutive characters. The value of CWT within the range of 12–32779 ETU. The time between transmitted characters is controlled by the programmable guard time in Transmitter Guard ETU Value (TX_GETU). However, the time between

Table continues on the next page...

Table 635. Transmitter features and requirements (continued)

Feature	Description
	the last byte in the transmit FIFO and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The EMVSIM transmitter provides a transmit FIFO threshold (TDTF) interrupt to signal the system when the expected number of characters has been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the TDTF interrupt and write new data to the transmit FIFO before the last character in the transmit FIFO has been sent.
Block waiting time	The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The EMVSIM transmitter provides two general purpose counters, either of which can be used to track the BWT. The BWT is purely determined by software response time to the transmit interrupts.
Block guard time	The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BGT is 22 ETU. The EMVSIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after XMT_EN = 1. The BGT is determined by the speed at which the software can react to an interrupt and enable the transmitter.
Error detection code	T = 1 cards can specify LRC or CRC error detection codes to be used. EMVSIM provides hardware support for both LRC and CRC operation.

69.3.5 LRC and CRC

EMVSIM also contains the 8-bit LRC block and 16-bit CRC block, which are common to the transmitter and receiver. Both the transmitter and the receiver can use the LRC and CRC blocks. They can insert the LRC or CRC into the byte stream or check the validity of the input stream using the LRC and CRC values received. Because transmit and receive operations are mutually exclusive, you can use LRC and CRC blocks that are shared by the transmitter and receiver.

69.3.5.1 LRC block

EMVSIM provides an 8-bit linear redundancy check (LRC) generator and checker. The block is provided for use with T = 1 smart cards that support LRC. This block can be enabled with CTRL[LRC_EN]. It performs an 8-bit exclusive-OR (XOR) on all received or transmitted characters.

At the end of the reception of a block of characters, the result is expected to be 00. If so, EMVSIM writes 1 to RX_STATUS[LRC_OK]. During transmission, the LRC block exclusive-ORs each character that is transmitted with the current value of the LRC. If CTRL[XMT_CRC_LRC] = 1, the transmitter automatically sends the LRC value as the final character when the transmit FIFO empties.

You can reset the LRC value in multiple ways:

- Writing 0 to CTRL[LRC_EN]
- Automatic reset by EMVSIM at the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when CTRL[XMT_CRC_LRC] = 0)
- By EMVSIM when writing 1 to CTRL[XMT_EN]

Here is a summary of the configuration needed to use the LRC block:

- Write 1 to CTRL[LRC_EN] to enable the LRC block.
- Use CTRL[XMT_CRC_LRC] to enable the transmission of the LRC character after the last character in the transmit FIFO has been sent.

69.3.5.2 CRC block

EMVSIM provides a 16-bit cyclic redundancy check (CRC) generator and checker. This block is provided for use with T = 1 smart cards that support CRC. The block can be enabled with CTRL[CRC_EN]. This block performs a polynomial-based check on all received or transmitted characters. The polynomial used for calculating the CRC is $G(x) = x^{16} + x^{12} + x^5 + 1$. EMVSIM initializes the CRC register by writing all 1s to it before the data is shifted in. You can configure the CRC block to compute using the ISO 7816 CRC method or the 16-bit CCITT CRC method by selecting the following fields:

- CTRL[CRC_IN_FLIP]
- CTRL[CRC_OUT_FLIP]
- CTRL[INV_CRC_VAL]

To select the CCITT CRC method, write 1 to:

- CTRL[CRC_IN_FLIP]
- CTRL[CRC_OUT_FLIP]
- CTRL[INV_CRC_VAL]

To select the normal CRC (ISO 7816 CRC) method, write:

- 0 to CRC_IN_FLIP
- 0 to CRC_OUT_FLIP
- 1 to INV_CRC_VAL

During transmission, the CRC block updates the current value of the CRC, using each character. If CTRL[XMT_CRC_LRC] = 1, the CRC value is automatically formatted according to the selected CRC method (CCITT or normal). The transmitter then sends it as the final two characters when the transmit FIFO empties. At the end of the reception of a block of characters, the residual from the CRC calculation is compared and EMVSIM writes 1 to RX_STATUS[CRC_OK].

NOTE

Any other value of CRC_IN_FLIP, CRC_OUT_FLIP, and INV_CRC_VAL causes the CRC value to be formatted accordingly; however, the CRC is not compliant with the standard CRC used in smart cards.

You can change the CRC value in multiple ways:

- Writing 0 to CTRL[CRC_EN]
- Automatic reset by EMVSIM at the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when CTRL[XMT_CRC_LRC] = 0)
- By EMVSIM when writing 1 to CTRL[XMT_EN]

Here is a summary of the configuration needed to use the CRC block:

- Write the correct values to CTRL[CRC_IN_FLIP], CTRL[CRC_OUT_FLIP], and CTRL[INV_CRC_VAL] to select the appropriate CRC method (CCITT or normal).
- Write 1 to CTRL[CRC_EN] to enable the CRC block.
- Write 1 to CTRL[XMT_CRC_LRC] to enable the transmission of the CRC characters after the last character in the transmit FIFO has been sent.

69.3.6 Message handling

EMVSIM has FIFOs on both the transmit side and the receive side for handling all message bytes.

69.3.6.1 Transmit FIFO

A 16-byte deep FIFO is implemented in the transmitter. The transmit FIFO can be written to by software, but an attempt to read it returns zeros. To manage clock synchronization, each write access has wait states inserted by EMVSIM, but this is transparent

to the software. You can write to the transmit FIFO when the transmitter is enabled; however, the transmitter does not start transmitting while the FIFO is empty. A write to the transmit FIFO initiates the transmit operation.

Software can store more bytes to the transmit FIFO until the programmed threshold is reached. You can enable DMA operation by writing 1 to CTRL[TX_DMA_EN]. EMVSIM asserts a DMA request that remains asserted until the number of bytes in the FIFO reaches the programmed threshold value (TX_THD[TDT[3:0]]).

The transmit FIFO can be flushed by writing 1 to CTRL[FLSH_TX]. A transmit NACK threshold error (TX_STATUS[TNTE]) halts the transmitter, and flushes the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The transmit data threshold flag (TX_STATUS[TDTF]) is not cleared by the FLSH_TX operation.

69.3.6.2 Receive FIFO

A 16-byte deep FIFO is implemented in the receiver. Because more than 16 bytes can be received in the FIFO, you must configure the software to ensure that it regularly reads the receive FIFO. You can read the receive FIFO by software via normal CPU accesses on interrupt (INT_MASK[RX_DATA_IM]) or via DMA by writing 1 to CTRL[RX_DMA_EN]. Software cannot write to this register. A write access may terminate in a transfer error.

The receiver loads the receive FIFO when it receives an error-free message byte with correct parity. The FIFO stores the message byte only. When the number of total bytes in the receive FIFO equals the programmed threshold value (RX_THD[RDT[3:0]]), EMVSIM writes 1 to RX_STATUS[RDTF]. An interrupt is asserted if INT_MASK[RDT_IM] = 0. If DMA access to RX_FIFO is enabled, then the DMA request to read the receive FIFO is asserted when the threshold flag is set and clears when all the bytes are read out from the receive FIFO.

The receive FIFO can be flushed by writing 1 to CTRL[FLSH_RX]. The flush operation resets the receiver except for the receiver control registers.

NOTE

Because the registers and the receiver logic are in different clock domains, you must configure the software to allow clock synchronization time when sampling RX_STATUS[RDTF]. One proposed method is:

- Read one byte at a time (from the receive FIFO) when RX_STATUS[RDTF] = 1.
- Allow a time of 3 bus clock cycles after the receive FIFO read.
- Check the status of RX_STATUS[RDTF].
- Repeat as necessary.

Alternatively, you could set the receiver threshold to be one more than is actually needed.

Receive FIFO Overflow Detection: When a byte is received, and the receive FIFO:

- Has not been read
- Already contains 16 bytes

then EMVSIM asserts a FIFO overflow error (EMVSIM writes 1 to RX_STATUS[RFO]). The received byte is discarded, leaving the FIFO with the first 16 bytes received.

If CTRL[ONACK] = 1, the transmitter generates a NACK for the overflow condition every time it receives a new byte that cannot be stored in the receive FIFO. EMVSIM continually sends NACKs until the receive FIFO is read or the NACK transmit threshold is reached.

69.3.7 Protocol timers

EMVSIM has several 16-bit and 32-bit timers that allow the software to perform protocol timing checks and measurements, such as:

- Transmit guard time
- Character wait time
- Block wait time

- Block guard time
- Work wait time

In addition to these timers, EMVSIM also contains two 32-bit general purpose counters that software can use for any specific time measurement application.

69.3.7.1 Transmit guard time

The transmit guard time is the number of ETUs inserted by the transmitter between each character that it transmits. You control the number of ETUs by programming Transmitter Guard ETU Value (TX_GETU). The guard time is measured as the time (in ETUs) between the STOP bits of the previous character and the start bit of the next character.

For a guard time programmed as FFh in TX_GETU[GETU], the transmitter inserts one STOP bit instead of two.

69.3.7.2 Character wait time

The EMVSIM receiver block includes a 16-bit counter that counts the number of bit times (ETUs) between received characters (that is, the time between start bits in consecutive characters that are received). When enabled, the CWT counter does not start counting until after the start bit(s) of a valid character have been received.

The counter is synchronized to the receive character bit positions so that an accurate count of the number of ETUs between characters can be made. The CWT has a 16-bit programmable comparator. Software can write the expected number of ETUs between characters to the comparator. If the time between characters exceeds this value, an interrupt flag is set and an interrupt is generated if the mask = 0.

You can configure the CWT like this:

- Program Character Wait Time Value (CWT_VAL) to the value that needs to be enforced.
- Activate the CWT function by writing 1 to CTRL[CWT_EN].
- Enable the CWT_ERR interrupt by writing 0 to INT_MASK[CWT_ERR_IM].
- If an interrupt occurs, then the software must read Receive Status (RX_STATUS) to determine if the error was caused by a CWT_ERR violation. The software must clear the CWT_ERR interrupt by writing 1 to RX_STATUS[CWT_ERR].
- Software can disable the CWT counter by writing 0 to CTRL[CWT_EN] at any time.

69.3.7.3 Block wait time and block guard time

A 32-bit timer is used to measure both block wait time (BWT) and block guard time (BGT). The BWT and BGT timer is used to measure the delay between:

- The leading edge of the last character of the block received by the card
- The leading edge of the first character of the next block sent by the card

If the time that has been measured is larger than the value in Block Wait Time Value (BWT_VAL), then INT_MASK[BWT_ERR_IM] automatically transitions to 1 and an interrupt is generated. If the time is less than the value in Block Guard Time Value (BGT_VAL), then INT_MASK[BGT_ERR_IM] automatically transitions to 1 and an interrupt is generated.

The block wait timer can be configured as follows:

- Program the BWT_VAL and BGT_VAL registers to the values that need to be enforced.
- Activate the block wait timer by writing 1 to CTRL[BWT_EN].
- Enable the BWT_ERR and BGT_ERR interrupts by writing 0 to INT_MASK[BWT_ERR_IM] and INT_MASK[BGT_ERR_IM].
- If an interrupt occurs, then the software must read the RX_STATUS register to determine if the error was caused by a BWT_ERR violation or by a BGT_ERR violation. The software should clear the interrupt status field by writing a 1 to either RX_STATUS[BWT_ERR] or RX_STATUS[BGT_ERR] respectively.
- Software can disable the BWT counter by writing 0 to CTRL[BWT_EN] at any time.

NOTE

BWT should be enabled after all bytes have been written to the transmit FIFO. This is necessary to avoid the condition where writes to the transmit FIFO are very slow (more than 1 ETU time apart) and if BWT is enabled earlier, then a BWT error might trigger falsely.

69.3.7.4 Work wait time

There is no separate counter to measure the work wait time (WWT). Work wait time is a combination of BWT and CWT. To measure WWT, software must configure both Character Wait Time Value (CWT_VAL) and Block Wait Time Value (BWT_VAL) to the value to be enforced. When measuring WWT the BGT timer is not used, so it should be masked (inactive) by INT_MASK[BGT_ERR_IM].

Here is the sequence to program EMVSIM to send and receive data while checking WWT:

- Program both the CWT_VAL and the BWT_VAL registers to the WWT (work wait time) value that needs to be enforced. Write 0 to Block Guard Time Value (BGT_VAL) (this is the default value).
- Activate both the CWT and BWT functions by writing 1 to CTRL[CWT_EN] and to CTRL[BWT_EN].
- Enable the CWT_ERR and BWT_ERR interrupts by writing 0 to INT_MASK[CWT_ERR_IM]] and INT_MASK_IM[BWT_ERR_IM].
- Program EMVSIM to enable either the receiver or the transmitter, depending on the application.
- If an interrupt occurs, then the software should consider it a WWT violation if any of these fields are asserted:
  - RX_STATUS[CWT_ERR]
  - INT_MASK[CWT_ERR_IM]
  - RX_STATUS[BWT_ERR]

  Program the software to clear the CWT_ERR or BWT_ERR interrupt by writing a 1 either to RX_STATUS[BWT_ERR] or to RX_STATUS[CWT_ERR].

69.3.7.5 General purpose counters

EMVSIM provides two 16-bit counters that it uses when timing events during smart card communication. The clock source for the counter is selectable between three sources:

- Card clock
- Receiver clock (16 times the ETU clock)
- ETU clock (for the transmitter)

Select the clock input with CLKCFG[GPCNT0_CLK_SEL] and CLKCFG[GPCNT1_CLK_SEL]. The counter is enabled as soon as you select the input clock. The counter starts immediately after the input clock begins running. You can use software to control the three input clock sources by using:

- PCSR[SCEN]
- CTRL[RCV_EN]
- CTRL[XMT_EN]

To run the counters from the card clock source, the following conditions must be met:

- CTRL[KILL_CLOCKS] = 0
- CTRL[RCV_EN] = 1 or CTRL[XMT_EN] = 1
- PCSR[SCEN] = 1
- CLKCFG[GPCNT0/1_CLK_SEL] = 01

The counter begins to count at the card clock rate as soon as these conditions are met.

To run the counters from the receive clock source, the following conditions must be met:

- CTRL[KILL_CLOCKS] = 0
- CTRL[RCV_EN] = 1 or CTRL[XMT_EN] = 1
- CLKCFG[GPCNT0/1_CLK_SEL] = 10

The counter begins to count at the receive clock rate as soon as these conditions are met.

To run the counters from the ETU (transmit) clock source the following conditions must be met:

- CTRL[KILL_CLOCKS] = 0
- CTRL[RCV_EN] = 1 or CTRL[XMT_EN] = 1
- CLKCFG[GPCNT0/1_CLK_SEL] = 11

The counter begins to count at the ETU clock rate as soon as these conditions are met.

You can reset the counters by writing 00 to GPCNT0/1_CLK_SEL. A 16-bit comparator value (GPCNT0/1_VAL register) allows the software to select a count value. When that count value is reached, then you can set the interrupt flags (TX_STATUS[KGPCNT0_TO] and TX_STATUS[KGPCNT1_TO]). An interrupt is generated if the mask (INT_MASK[GPCNT0_IM] or INT_MASK[GPCNT1_IM]) = 0.

Use the following sequence to configure both general purpose timers:

1. Program the counter comparator using the GPCNT0/1_VAL register.
2. Enable the conditions described above using the CTRL register.
3. Select a clock source for the general purpose counter using the CLKCFG register.
4. Enable interrupts in the INT_MASK register.

69.3.8 Answer-to-reset (ATR) detection

The first communication that occurs between the smart card and EMVSIM is a block of data that the smart card sends to EMVSIM. This communication happens after the card is powered up and the card reset is removed (see Smart card interface and control). This block of data is called the answer-to-reset (ATR). To receive the ATR, you must configure EMVSIM for 12 ETU character reception. According to the ISO 7816-3 spec, both T = 0 and T = 1 cards communicate initially using 12 ETU character durations.

For more information about ATR detect configuration, see the Configuration for ATR detect section.

69.3.9 Clocks

The key functional blocks in EMVSIM that control the interactions with the smart card are:

- Clocking
- Transmitter
- Receiver

These blocks function on a clock that is different from, and asynchronous to, the register read/write clock. In order for the transmitter and receiver to function properly, the respective configuration must be complete before you enable the transmitter or receiver.

The clocking module generates the following clocks:

- Card clock (output to smart card)
  - The prescaler (CLKCFG[CLK_PRSC]) generates the card clock in the range of 1–5 MHz with near 50% duty cycle.
- Internal clocks for the transmitter and receiver (internal clocks control the ETU period and the sampling and driving of data on the IO line)
  - The receive operation uses an internal oversampled clock that oversamples the received input at 16 times the ETU rate.

— The baud divider (Baud Rate Divisor (DIVISOR)) generates the internal ETU clocks for transmitter and receiver.

69.3.10 Modes

EMVSIM has two modes of operation:

- Stop mode: All EMVSIM clocks shut down in Stop mode except the smart card clock. It is still active when CTRL[STOP_EN] = 1.
- Doze mode: If CTRL[DOZE_EN] = 1, all EMVSIM clocks shut down when the transmit FIFO is empty. Otherwise, a Doze instruction has no effect on EMVSIM.

69.3.11 Reset

CTRL[SW_RST] is used to reset the entire EMVSIM module. The result is the same as a hardware reset. There is also PCSR[SRST], which is used to control the state of the reset line to the smart card. It can be forced low by hardware during the auto power-down sequence.

69.3.12 Interrupts

EMVSIM has one interrupt output. It is generated by any of the following conditions:

- Parity error flag
- Block guard time error flag
- Block wait time error flag
- Received NACK threshold error flag
- Character wait time error flag
- Receive data threshold interrupt flag
- Receive data interrupt flag
- Receive FIFO overflow flag
- General purpose counter 1 timeout flag
- General purpose counter 0 timeout flag
- Transmit data threshold flag
- Transmit FIFO full flag
- Transmit complete flag
- Early transmit complete flag
- Transmit FIFO empty flag
- Transmit NACK threshold error flag

69.3.13 DMA

The EMVSIM DMA supports transferring data from and to FIFOs.

Software can enable DMA operation by writing 1 to CTRL[TX_DMA_EN]. When this field = 1, EMVSIM asserts a DMA request that remains asserted until the number of bytes in the FIFO reaches the programmed threshold value (TX_THD[TDT]).

EMVSIM enables DMA read access when CTRL[RX_DMA_EN] = 1. When this field = 1, EMVSIM asserts a DMA request to read the Rx FIFO when the threshold flag (TX_STATUS[TDTF]) = 1. EMVSIM writes 0 to TX_STATUS[TDTF] when all the bytes have been read from the Rx FIFO.

69.4 External signals

EMVSIM interfaces to the external card (smart card) through:

- Card clock
- Card reset
- Card V_{CC} enable
- I/O pins

The transmitter output from the module, and the receiver input to the module, are connected to the same I/O pin and are used in open-drain configuration.

NOTE

You must connect a pullup resistor to the I/O pin on board to allow proper functioning.

The following table shows the user-accessible signals available on EMVSIM. See the chip-level specification to find out which signals are connected to the external pins.

Table 636. External signals

Signal	Description	I/O
CLK	Card clock—clock to smart card	O
RST	Card reset—reset signal to smart card	O
SVEN	Card power enable—Controls power to smart card	O
IO	Card data line—Bidirectional data line	I/O
PD	Card presence detect—Indicates presence or removal of card	I

69.5 Initialization

The first thing to do is to configure the interface signals, allowing transmit or receive operation to begin. The following steps describe a typical way to enable the smart card:

1. Configure the prescaler by programming the selected value in CLKCFG[CLK_PRSC] to generate the card clock. The card clock frequency is calculated as f_{SIM} / (CLKCFG[CLK_PRSC]) (where f_{SIM} is the EMVSIM clock frequency).

   You must set the prescaler value so that the maximum card clock frequency specifications, as stated in EMV and ISO-7816 specifications, are not exceeded.
2. Configure DIVISOR[DIVISOR_VALUE]. This value determines the ETU period that the transmitter or receiver uses.
3. Enable power to the smart card by writing 1 to PCSR[SVCC_EN].
4. Enable the smart card clock by writing 1 to PCSR[SCEN].
5. Release the smart card reset by writing 1 to PCSR[SRST].
6. Program the power-down functionality, that is, whether auto power-down of the smart card is required or not.
7. There are two ways to select the data format type for EMVSIM. Use only one of them.
   - Select the data format (inverse or direct) by writing the appropriate value to CTRL[IC].
   - Write 1 to CTRL[ICM] to allow automatic detection of the initial character. EMVSIM updates CTRL[IC] to the selected data format when a valid initial character has been detected.
8. Configure EMVSIM to run in Stop mode by writing 1 to CTRL[STOP_EN], and in Doze mode by writing 1 to CTRL[DOZE_EN], as needed.
9. Enable DMA request generation for the transmitter or receiver as appropriate.

After the configuration is complete, you can configure the receiver or transmitter, depending on the operation to be carried out on the smart card.

NOTE

You must not change CTRL[XMT_EN] and CTRL[RCV_EN] in the same write cycle. Software should attempt separate write accesses to the CTRL register for updating these fields, one by one. For example, if XMT_EN = 1 and RCV_EN = 0, then enable the receiver like this: the first write to the CTRL register should make XMT_EN = 0, and the second write to the CTRL register should make RCV_EN = 1.

The transmit and receive operations are mutually exclusive. You must not enable them together.

69.5.1 Configuring the receiver

Here is a list of configurations that you need to perform (after Initialization) in order to configure the EMVSIM receiver for operation:

1. Enable NACK generation capability.
   - To generate NACK on parity errors and invalid initial characters, write 1 to CTRL[ANACK].
   - To generate NACK on receiver FIFO overflow error, write 1 to CTRL[ONACK].
2. Configure thresholds.
   - Configure the threshold for NACK generation by writing the appropriate value to RX_THD[RNCK_THD].
   - Configure the threshold for data by writing the appropriate value to RX_THD[RDT].

   These are the thresholds at which the RTE and RDTF flags are set, respectively.
   - If auto power-down is needed on RTE flag assertion, then write 1 to PCSR[SAPD] to initiate the smart card power-down.
3. Configure the timeout values for the CWT Counter, the Block Wait Timer Counter, and the Block Guard Time Counter. [Identify topics to link to for reference.]
4. Enable necessary interrupts by writing 0 to the appropriate fields in Clock Configuration (CLKCFG).
5. Configure the general purpose counter as needed and select the correct clock source (CLKCFG[GPCNT0_CLK_SEL] or CLKCFG[GPCNT1_CLK_SEL]).
6. Enable the receiver by writing 1 to CTRL[RCV_EN].
7. The receiver enters Initial Character Detection mode if CTRL[ICM] = 1 or enters normal receive mode if CTRL[ICM] = 0.
8. Enable either LRC block or CRC block.

69.5.2 Configuring the transmitter

Here is a list of the configurations that you need to perform (after the Initialization step) in order to configure the EMVSIM transmitter for operation:

1. Select a retransmission threshold for NACKed characters in Transmitter Threshold (TX_THD). This is the threshold at which TX_STATUS[TNTE] is set by using TX_THD[TNCK_THD].
2. Configure the guard time between the transmitted characters by writing the appropriate value to TX_GETU[GETU].
3. Select the appropriate data threshold in TX_THD.
4. Enable necessary interrupts by writing 0 to the appropriate fields in Interrupt Mask (INT_MASK).
5. Configure the general purpose counter as needed and select the correct clock source (CLKCFG[GPCNT0_CLK_SEL] or CLKCFG[GPCNT1_CLK_SEL]).
6. Enable the transmitter by writing 1 to CTRL[XMT_EN].
7. Enable LRC and CRC as needed.[more detail]

8. Program bytes to be transmitted into the transmit FIFO. If DMA is enabled, then a DMA request is asserted automatically.

NOTE

You can write to the transmit FIFO after the transmitter is enabled (CTRL[XMT_EN] = 1). Writing to the transmit FIFO when the transmitter is disabled is not successful and write access is ignored.

69.6 Application information

69.6.1 Powering up

The first step in communicating with a smart card is to provide the card with power and a clock signal. After the card is detected as present (using the presence detect features, or by some other method), you should power up the smart card according to the power-up sequence provided in the ISO 7816-3 specification.

1. Apply voltage to the smart card by writing 1 to PCSR[SVCC_EN].
2. Select the appropriate clock frequency for the smart card by programming Clock Configuration (CLKCFG) and Baud Rate Divisor (DIVISOR).
3. Enable the clock to the smart card by writing 1 to PCSR[SCEN].
4. The card exits reset state by writing 1 to PCSR[SRST].

69.6.2 Automatic power-down

Smart cards require a proper sequence of operations to power down. The power-down sequence as per ISO 7816 specifications is:

1. Smart card reset (RST) pin transitions from high to low.
2. Smart card clock (CLK) is turned off to low.
3. Smart card IO (IO) is transitioned from high impedance to low.
4. Smart card voltage enable (SVCC_EN) is turned off (pin driven low).

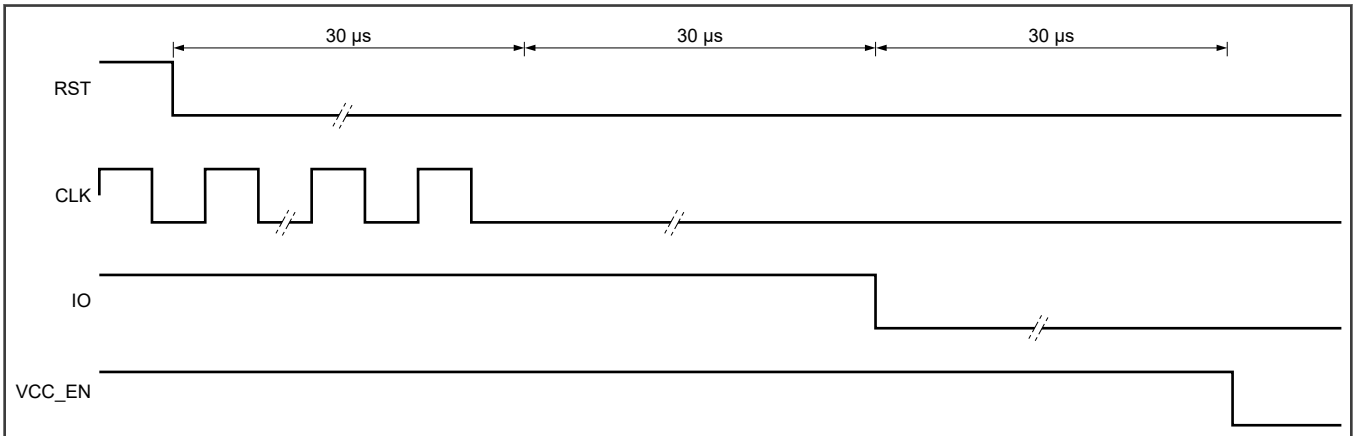


Figure 412. Auto power-down sequence

These operations can be done manually by software which writes to the appropriate fields (PCSR[SCEN], PCSR[SRST], and PCSR[SVCC_EN]), or automatically by hardware under the following conditions:

- You write 1 to PCSR[SPD]
- Assertion of RX_STATUS[RTE]
- Assertion of PCSR[SPDIF] when configured to detect card removal

NOTE

PCSR[SPDES] determines which edge transition of the smart card presence detect pin (PD) is used for smart card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the edge specified by PCSR[SPDES] causes EMVSIM to do the following:

- PCSR[SPDIF] automatically transitions to 1
- If PCSR[SPDIM] = 0, generate an interrupt to the CPU
- If PCSR[SAPD] = 1, begin an auto power-down sequence

If smart card insertion is expected, you can write 0 to PCSR[SAPD] low to avoid the auto power-down sequence from falsely triggering. You can use PCSR[SPDP] to determine the current state of the smart card presence detect pin.

The auto power-down operation requires a low-frequency clock (usually a 32 kHz clock source) to be active on the device.

The auto power-down sequence writes 0 to the following fields upon power-down sequence completion:

1. PCSR[SPD]
2. PCSR[SCEN]
3. PCSR[SRST]
4. PCSR[SVCC_EN]
5. CTRL[XMT_EN] and CTRL[RCV_EN]

69.6.3 Configuration for ATR detect

The following steps provide a suggested approach for configuring EMVSIM to receive the ATR.

1. Write 0 to CTRL[RCVR_11].
2. Write 1 to CTRL[ANACK] to enable NACK generation. The ISO 7816-3 spec allows the EMVSIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

   NOTE

   The Europay MasterCard and VISA (EMV) cards are similar to T = 1, but do not allow EMVSIM to NACK during the initial communication.

3. Enable RDTF and RFO interrupts by writing 0 to INT_MASK[RDT_IM] and INT_MASK[RFO_IM]. These interrupts are enabled to notify the software when characters are received. You can set a threshold for the number of characters to receive before generating an interrupt.
4. Set the desired threshold for received characters by writing RX_THD[RDT].

You must set up EMVSIM to perform in initial character mode during ATR detection. This causes the hardware to identify the first valid character sent during the ATR as an initial character. This character automatically configures the hardware for the data convention used by the smart card.

Set Initial Character mode by writing 1 to CTRL[ICM].

The ISO 7816-3 spec requires that smart cards meet certain timing restrictions. One of these is the time from the deassertion of the card reset to the beginning of the ATR sequence. The general purpose counter of EMVSIM can be used to verify that the smart card begins its ATR within the 400–40,000 card clock cycle range.

1. Write the general purpose counter timeout value with 9C40h by programming General Purpose Counter 0 Timeout Value (GPCNT0_VAL) and General Purpose Counter 0 Timeout Value (GPCNT0_VAL).
2. Enable the general purpose counter interrupt by writing 0 to INT_MASK[GPCNT0_IM] and INT_MASK[GPCNT1_IM].
3. Enable the general purpose counter by writing 01 to CLKCFG[GPCNT0_CLK_SEL] and to CLKCFG[GPCNT1_CLK_SEL] so that the card clock is used for counting.

The ISO 7816-3 spec states that the maximum allowed time between two characters during the ATR is 9600 ETUs (initial waiting time). You must set up the CWT counter to detect any errors for this condition.

1. Set the CWT counter comparator to 9600 using Character Wait Time Value (CWT_VAL).
2. Enable the CWT counter interrupt by writing 0 to INT_MASK[CWT_ERR_IM].
3. Enable the CWT counter by writing 1 to CTRL[CWT_EN].

The last step in preparing for ATR reception is to enable the receiver.

Write 1 to CTRL[RCV_EN].

EMVSIM generates interrupts after a threshold number of characters is received. The software must react to these interrupts and read the characters from the receive FIFO (RX_BUF) until the complete ATR has been received. If a general purpose counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 spec (see Smart card interface and control). Otherwise, when a valid ATR is received, the software determines from the ATR information what are the specific characteristics for this card (see the ISO 7816-3 spec for details).

69.6.3.1 Programming considerations for T = 0 smart cards

For type T = 0 cards, you must program the software to adjust the following parameters according to the information in the ATR:

- Adjust the baud rate by changing the values in Baud Rate Divisor (DIVISOR) based on the values of 'F' and 'D' received in the ATR.
- Adjust the guard time between characters by changing the value of TX_GETU[GETU].
- Adjust NACK capability by modifying the values of CTRL[ONACK] and CTRL[ANACK].
- Adjust the stop clock polarity by modifying the value of PCSR[SCSP].
- Adjust the level of transmit NACK retransmissions allowed by modifying the value of TX_THD[TNCK_THD].
- Adjust the level for the receive NACK threshold by modifying RX_THD[RNCK_THD].

If a negotiation with the smart card is needed, the software sends a PPS response to the smart card. To send the response, the following steps should be performed:

- Set the transmit FIFO threshold level by writing to TX_THD[TDT].
- Clear all transmit interrupt flags in Transmitter Status (TX_STATUS) by writing 1 to every bit in the register.
- Enable the transmit interrupts by writing 0 to the mask fields in Interrupt Mask (INT_MASK). If more than 16 characters are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the smart card.
- Enable the transmitter by writing 1 to CTRL[XMT_EN].
- Write the characters to be sent as response (maximum of 16) to the transmit FIFO using Transmit Data Buffer (TX_BUF).

At this point, EMVSIM transmits the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt is set when the number of characters remaining in the FIFO equals the threshold. The software can then write an additional number of characters to be sent without interrupting transmission to the smart card.

After the transmission is complete, EMVSIM should be completely configured for standard operation with the T = 0 smart card. The software can continue to service RDTF interrupts for received characters, and TDTF interrupts for transmitted characters.

69.6.3.2 Programming considerations for T = 1 smart cards

For type T = 1 cards, you must program the software to adjust the following parameters according to the information in the ATR:

- Adjust the baud rate by changing the values in Baud Rate Divisor (DIVISOR) based on the values of 'F' and 'D' received in the ATR.
- Adjust the guard time between characters by changing the value of TX_GETU[GETU]. Writing FFh to TX_GETU[GETU] configures the SIM transmitter for 11 ETU transmissions.

- Disable NACK capability by writing 0 to CTRL[ONACK] and CTRL[ANACK]. T = 1 cards do not allow NACKs.
- Adjust the stop clock polarity by modifying the value of PCSR[SCSP].
- Set the character wait time counter timeout value to the value specified in the ATR by using Character Wait Time Value (CWT_VAL).
- Enable the character wait time counter interrupt by writing 0 to INT_MASK[CWT_ERR_IM].
- Enable the character wait time counter by writing 1 to CTRL[CWT_EN].
- Enable CRC or LRC error checking according to the ATR information by writing 1 to either CTRL[CRC_EN] or CTRL[LRC_EN]. These fields must never be equal to 1 at the same time.

For T = 1 cards, the ATR is sent using a T = 0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the smart card is needed, the software sends a PPS response to the smart card. Otherwise, the protocol is initiated with a block transfer from EMVSIM. In order to send the response or the first block, the following steps must be performed:

- Set the transmit FIFO threshold level by writing to TX_THD[TDT].
- Clear all transmit interrupt flags in Transmitter Status (TX_STATUS) by writing 1 to every bit in the register.
- Enable the transmit interrupts by writing 0 to the mask bits in Interrupt Mask (INT_MASK). If more than 16 characters are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the smart card.
- Enable the transmission of the error checking characters (LRC or CRC) by writing 1 to CTRL[XMT_CRC_LRC].

NOTE

If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is complete. If so, do not write 1 to CTRL[XMT_CRC_LRC] during the PPS exchange.

- Enable the transmitter by writing 1 to CTRL[XMT_EN].
- Write the characters to be sent as response (maximum of 16) to the transmit FIFO using Transmit Data Buffer (TX_BUF).

At this point, EMVSIM transmits the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt is enabled when the number of characters remaining in the FIFO equals the threshold. The software can then write an additional number of characters to be sent without interrupting transmission to the smart card.

After the transmission is complete, EMVSIM should be completely configured for standard operation with the T = 1 smart card. The software can continue to service RDTF interrupts for received characters, and TDTF interrupts for transmitted characters.

69.7 Memory map and registers

69.7.1 EMVSIM register descriptions

EMVSIM memory consists of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. A write access to reserved locations generates a transfer error. A read access to reserved locations also generates a transfer error, and the read data bus shows all zeroes. The module does not check for the correctness of programmed values in the registers—therefore, software must ensure that correct values are written.

69.7.1.1 EMVSIM memory map

EMVSIM0 base address: 4010_3000h

EMVSIM1 base address: 4010_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VER_ID)	32	R	0000_0000h
4h	Parameters (PARAM)	32	R	0000_1010h
8h	Clock Configuration (CLKCFG)	32	RW	0000_0000h
Ch	Baud Rate Divisor (DIVISOR)	32	RW	0000_0174h
10h	Control (CTRL)	32	RW	0100_0006h
14h	Interrupt Mask (INT_MASK)	32	RW	0000_FFFFh
18h	Receiver Threshold (RX_THD)	32	RW	0000_0001h
1Ch	Transmitter Threshold (TX_THD)	32	RW	0000_000Fh
20h	Receive Status (RX_STATUS)	32	RW	0000_0000h
24h	Transmitter Status (TX_STATUS)	32	RW	0000_00B8h
28h	Port Control and Status (PCSR)	32	RW	0100_0000h
2Ch	Receive Data Read Buffer (RX_BUF)	32	R	0000_0000h
30h	Transmit Data Buffer (TX_BUF)	32	RW	0000_0000h
34h	Transmitter Guard ETU Value (TX_GETU)	32	RW	0000_0000h
38h	Character Wait Time Value (CWT_VAL)	32	RW	0000_FFFFh
3Ch	Block Wait Time Value (BWT_VAL)	32	RW	FFFF_FFFFh
40h	Block Guard Time Value (BGT_VAL)	32	RW	0000_0000h
44h	General Purpose Counter 0 Timeout Value (GPCNT0_VAL)	32	RW	0000_FFFFh
48h	General Purpose Counter 1 Timeout Value (GPCNT1_VAL)	32	RW	0000_FFFFh

69.7.1.2 Version ID (VER_ID)

Offset

Register	Offset
VER_ID	0h

Function
Contains the version ID of EMVSIM.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 VER	Version ID Contains the EMVSIM version number used on the chip. 31:16 — Major version (example: 01.00) 15:0 — Minor version (example: 01.00)

69.7.1.3 Parameters (PARAM)

Offset

Register	Offset
PARAM	4h

Function

Provides details on the parameter settings used when including this module in the chip.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TX_FIFO_DEPTH								RX_FIFO_DEPTH							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-8 TX_FIFO_DEPTH	Transmit FIFO Depth Contains value of parameter for transmit FIFO depth in bytes.
7-0 RX_FIFO_DEPTH	Receive FIFO Depth Contains value of parameter for receive FIFO depth in bytes.

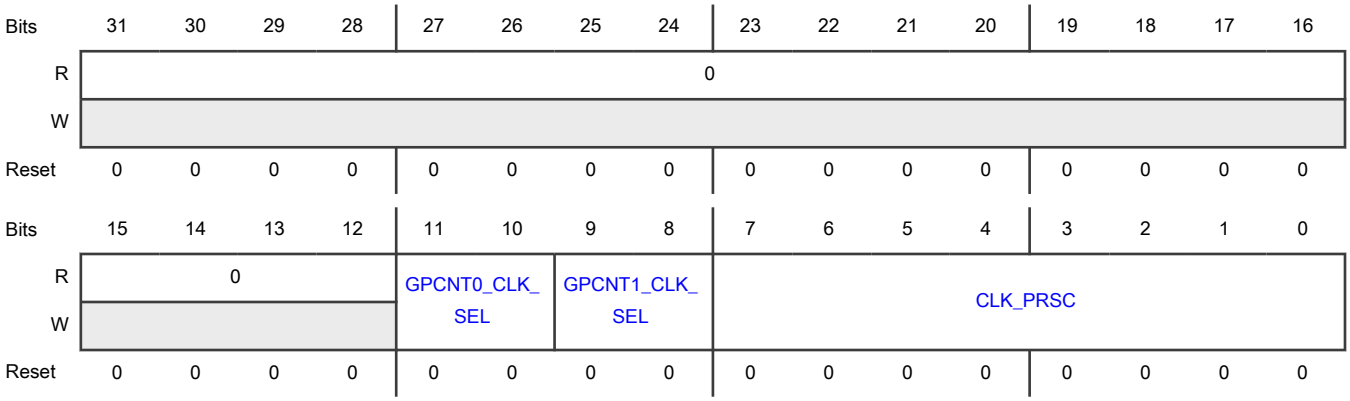
69.7.1.4 Clock Configuration (CLKCFG)

Offset

Register	Offset
CLKCFG	8h

Function
Provides configuration details to enable the correct clock frequency of EMVSIM clocks.

Diagram



Fields

Field	Function
31-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-10 GPCNT0_CLK_SEL	General Purpose Counter 0 Clock Select Selects which clock source is used by EMVSIM general purpose counter 0. The only way to reset the counter is to write 0 to this field. The counter begins counting as soon as the clock input is selected and the clocks are active. These input clocks are enabled through other EMVSIM fields (CTRL[KILL_CLOCKS], CTRL[RCV_EN], and CTRL[XMT_EN]). The counter is active if CTRL[RCV_EN] = 1 or CTRL[XMT_EN] = 1. 00b - Disable/reset 01b - Card clock 10b - Receive clock 11b - ETU clock (transmit clock)
9-8 GPCNT1_CLK_SEL	General Purpose Counter 1 Clock Select Selects which clock source is used by EMVSIM general purpose counter 1. The only way to reset the counter is to write 0 to this field. The counter begins counting as soon as the clock input is selected and the clocks are active. These input clocks are enabled through other EMVSIM fields (CTRL[KILL_CLOCKS], CTRL[RCV_EN], and CTRL[XMT_EN]). The counter is active if CTRL[RCV_EN] = 1 or CTRL[XMT_EN] = 1. 00b - Disable/reset 01b - Card clock 10b - Receive clock 11b - ETU clock (transmit clock)
7-0 CLK_PRSC	Clock Prescaler Value Contains the prescaler value that you can use to select the card clock frequency. The card clock frequency is the EMVSIM protocol (or logic) clock divided by this prescaler value. Update the prescaler value only when card clock is disabled and the transmitter and receiver are not operational. When this field = 0 or 1, EMVSIM uses the maximum prescaler value (FFh). For all other values of this field, the prescaler value is the value of this field.

69.7.1.5 Baud Rate Divisor (DIVISOR)

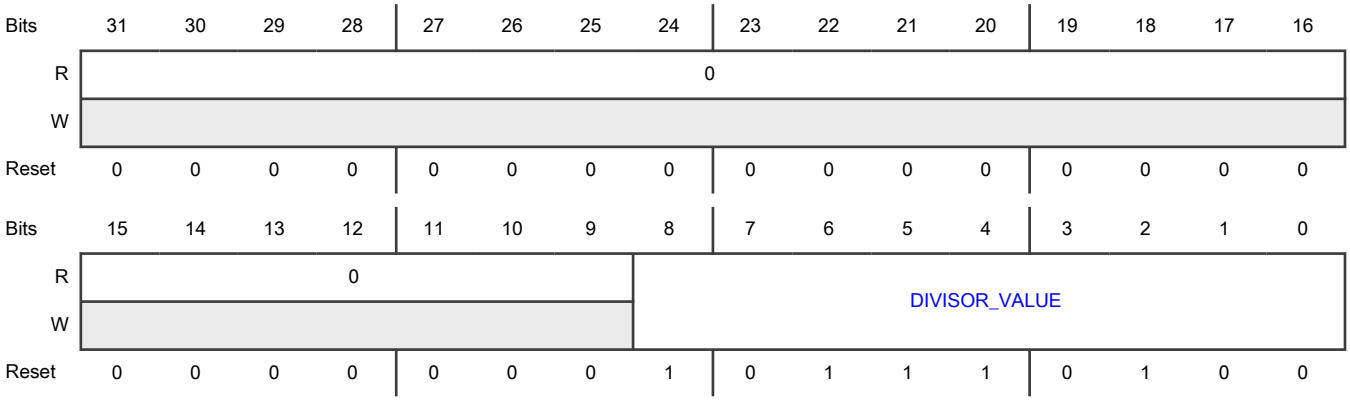
Offset

Register	Offset
DIVISOR	Ch

Function

Configures the divisor value to generate the baud-rate clock. The baud-rate clock drives the card clock and also generates the transmit and receive clocks and their respective ETUs.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-0 DIVISOR_VALUE	Divisor (F/D) Value Contains the divisor value. The value written to this register is used to generate the ETU bit period that is used by the transmitter and receiver. The divisor value is the integer result of the F/D value. 1 ETU = F / D x 1 / f • The ETU is the elementary time unit, the nominal time to transmit a bit (0 or 1). • F is the clock rate conversion integer. • D is the baud rate adjustment integer. • f is the frequency of the communication clock used between the controller and the smart card. You can change the value in this register when there is no active transaction with the smart card. When you program this field, the fractional part of F/D must be rounded off to the nearest integer. 0_0000_0000b-0_0000_0100b - Invalid. As per ISO 7816 specification, the minimum value of F/D is 5. 0_0000_0101b-0_1111_1111b - Divisor value F/D. For example, 101110100b = the divisor value for F = 372 and D = 1.

69.7.1.6 Control (CTRL)

Offset

Register	Offset
CTRL	10h

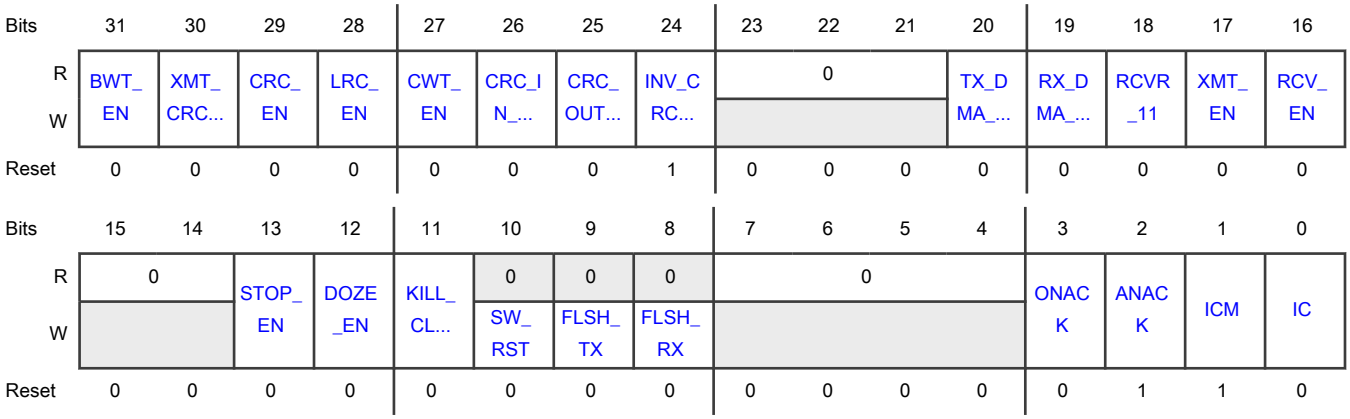
Function

Provides EMVSIM configuration details.

NOTE

You must not change XMT_EN and RCV_EN fields in the same write cycle. Program the software to write separate accesses to them, performing each update one at a time.

Diagram



Fields

Field	Function
31 BWT_EN	Block Wait Time Counter Enable Enables the BWT and BGT functions. You can then individually select the BWT and BGT functions using the interrupt mask. 0b - Disable 1b - Enable
30 XMT_CRC_LRC	Transmit CRC or LRC Enable Specifies whether to transmit the redundancy checking data (LRC or CRC) at the end of a transmission (that is, when the FIFO becomes empty). 0b - Do not transmit 1b - Transmit
29 CRC_EN	CRC Enable Enables 16-bit cyclic redundancy checking, using calculation of the 16-bit CRC value for both receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and is reflected in RX_STATUS[CRC_OK]. Writing 0 to this field resets the current CRC residual value in the EMVSIM hardware. 0b - Disable 1b - Enable
28 LRC_EN	LRC Enable Enables 8-bit linear redundancy checking, using calculation of the 8-bit LRC value for both receiver and transmitter. The result of the calculation is continuously compared to zero and is reflected in RX_STATUS[LRC_OK]. Writing 0 to this field resets the current LRC value in the EMVSIM hardware.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
27 CWT_EN	CWT Counter Enable Enables the character wait time counter. Writing 0 to this field resets the counter to zero. 0b - Disable 1b - Enable
26 CRC_IN_FLIP	CRC Input Byte's Bit Reversal Or Flip Control Controls whether the bits in the CRC input byte are reversed before CRC calculation. When this field = 1, the bits within the input byte for CRC are reversed from 7:0 to 0:7. When this field = 0, then the bits in the input byte are not reversed (that is, 7:0 remains 7:0). For CCITT CRC calculation, this field must = 1. For any other CRC using the same polynomial, write to this field accordingly. 0b - Not reversed 1b - Reversed
25 CRC_OUT_FLIP	CRC Output Value Bit Reversal Or Flip Control Controls whether the bits in the CRC output bytes are reversed. When this field = 1, the bits in the two bytes used for CRC output change from 15:0 to {8:15,0:7}. When this field = 0, then the bits in the output bytes are not reversed. For CCITT CRC calculation, this field must = 1. For any other CRC using the same polynomial, write to this field accordingly. 0b - Not reversed 1b - Reversed
24 INV_CRC_VAL	Invert CRC Output Value Bits Controls whether the bits within the CRC Output value are inverted (1's complement). For CCITT CRC calculation, you must write 1 to this field. For any other CRC using the same polynomial, write to this field accordingly. 0b - Not inverted 1b - Inverted
23-21 —	Reserved
20 TX_DMA_EN	Transmit DMA Enable Enables assertion of a DMA write request when the transmit FIFO is empty. The request is held asserted until the transmit FIFO reaches the programmed data threshold value. A transmit data threshold interrupt is not generated when this field is asserted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not asserted 1b - Asserted
19 RX_DMA_EN	Receive DMA Enable Enables assertion of a DMA read request when the receive FIFO reaches the programmed data threshold value. The request is held asserted until all the programmed threshold bytes are read out. A receiver data interrupt is not generated when this field is asserted. 0b - Not asserted 1b - Asserted
18 RCVR_11	Receiver 11 ETU Mode Enable Configures the EMVSIM receiver for 11 ETU operation (that is, 1 stop bit). This field is provided for support of T = 1 cards. If this field = 0, the receiver is configured for 12 ETU operation mode. If this field = 1, the receiver is configured for 11 ETU operation mode. 0b - 12 ETU operation 1b - 11 ETU operation
17 XMT_EN	Transmitter Enable Enables the EMVSIM transmitter block. You can set it to 0 during the auto–power-down sequence. **Note:** Writing 1 to this field resets the CRC and LRC values. 0b - Disable 1b - Enable
16 RCV_EN	Receiver Enable Enables the EMVSIM receiver block. After the transmitter has completed its operation, you must use this field to enable the receiver. You can write 0 to it during the auto–power-down sequence. 0b - Disable 1b - Enable
15-14 —	Reserved
13 STOP_EN	STOP Enable Configures the operation of EMVSIM when a processor Stop instruction is executed. This field provides support for smart cards that do not allow the smart card clock to be stopped while power is applied. If this field = 0, a Stop instruction shuts down all EMVSIM clocks. If this field = 1, a Stop instruction shuts down all clocks except the smart card clock. 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 DOZE_EN	Doze Enable Configures the operation of EMVSIM when a processor Doze instruction is executed. If this field = 0, a Doze instruction gates all internal EMVSIM clocks as well as the smart card clock when the transmit FIFO is empty. If this field = 1, a Doze instruction has no effect on EMVSIM. 0b - Disable 1b - Enable
11 KILL_CLOCKS	Kill Internal Clocks Enables the clock input to EMVSIM. This field gates all clocks, including the smart card clock, regardless of the state of the Stop bit described above. This field has no effect on the register read/write clock. Only the EMVSIM logic clock is gated. 0b - Enable 1b - Disable
10 SW_RST	Software Reset Resets the entire EMVSIM. This action has the same result as a hardware reset. This field is self-clearing and always reads 0. You must allow a minimum of four protocol clock cycles before attempting to access EMVSIM after a software reset. 0b - Normal 1b - Reset
9 FLSH_TX	Flush Transmitter Operates as an EMVSIM transmitter reset. The receive portion of EMVSIM is not affected. This field is self-clearing and always reads 0. 0b - Normal 1b - Reset
8 FLSH_RX	Flush Receiver Operates as an EMVSIM receiver reset. The transmit portion of EMVSIM is not affected. This field is self-clearing and always reads 0. 0b - Normal 1b - Reset
7-4 —	Reserved
3 ONACK	Overrun NACK Enable Enables NACK generation when the receive FIFO is full and another message is ready for writing into the FIFO.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
2 ANACK	Auto NACK Enable Enables NACK generation for parity errors in received messages or when invalid initial characters are received in ICM mode. 0b - Disable 1b - Enable
1 ICM	Initial Character Mode Enables initial character mode. Hardware automatically clears this field after a valid initial character is received. 0b - Disable 1b - Enable
0 IC	Inverse Convention Configures EMVSIM to use either inverse convention or direct convention for its data format. You can control this field by software, but hardware normally sets it as a result of the interpretation of the initial character when in ICM mode. 0b - Direct 1b - Inverse

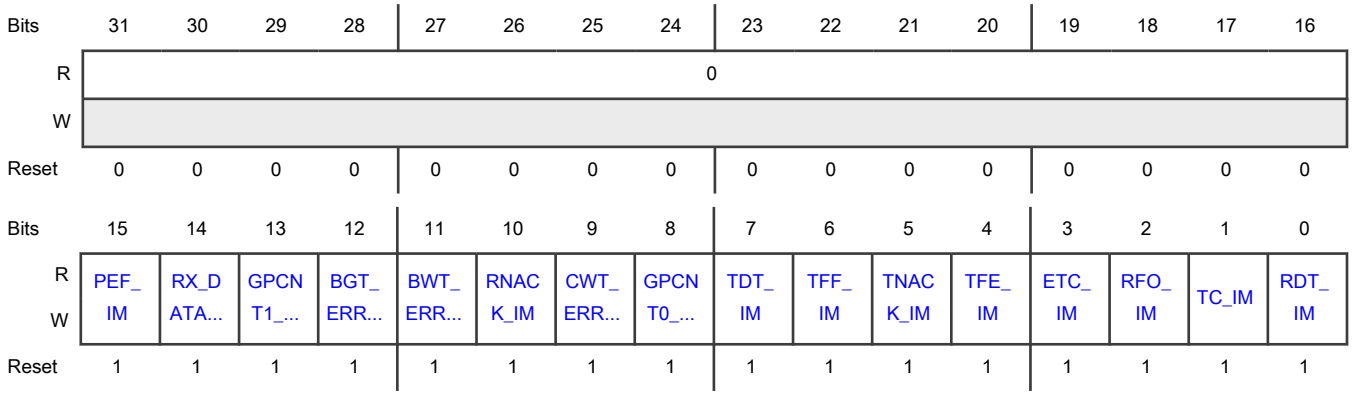
69.7.1.7 Interrupt Mask (INT_MASK)

Offset

Register	Offset
INT_MASK	14h

Function
Provides interrupt masking configuration details.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 PEF_IM	Parity Error Interrupt Mask Enables RX_STATUS[PEF] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
14 RX_DATA_IM	Receive Data Interrupt Mask Enables RX_STATUS[RX_DATA] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
13 GPCNT1_IM	General Purpose Counter 1 Timeout Interrupt Mask Enables TX_STATUS[GPCNT1_TO] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
12 BGT_ERR_IM	Block Guard Time Error Interrupt Enables RX_STATUS[BGT_ERR] flag to generate EMVSIM interrupts. 0b - Enable 1b - Masked
11 BWT_ERR_IM	Block Wait Time Error Interrupt Mask Enables RX_STATUS[BWT_ERR] to generate EMVSIM interrupts. 0b - Enable 1b - Masked

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 RNACK_IM	Receiver NACK Threshold Interrupt Mask Enables RX_STATUS[RTE] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
9 CWT_ERR_IM	Character Wait Time Error Interrupt Mask Enables RX_STATUS[CWT_ERR] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
8 GPCNT0_IM	General Purpose Timer 0 Timeout Interrupt Mask Enable/s TX_STATUS[GPCNT0_TO] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
7 TDT_IM	Transmit Data Threshold Interrupt Mask Enables TX_STATUS[TDTF] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
6 TFF_IM	Transmit FIFO Full Interrupt Mask Enables TX_STATUS[TFF] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
5 TNACK_IM	Transmit NACK Threshold Interrupt Mask Enables TX_STATUS[TNTE] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
4 TFE_IM	Transmit FIFO Empty Interrupt Mask Enables TX_STATUS[TFE] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
3 ETC_IM	Early Transmit Complete Interrupt Mask Enables TX_STATUS[ETCF] to generate EMVSIM interrupts. 0b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Masked
2 RFO_IM	Receive FIFO Overflow Interrupt Mask Enables RX_STATUS[RFO] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
1 TC_IM	Transmit Complete Interrupt Mask Enables RX_STATUS[TCF] to generate EMVSIM interrupts. 0b - Enable 1b - Masked
0 RDT_IM	Receive Data Threshold Interrupt Mask Enables RX_STATUS[RDTF] to generate EMVSIM interrupts. 0b - Enable 1b - Masked

69.7.1.8 Receiver Threshold (RX_THD)

Offset

Register	Offset
RX_THD	18h

Function
Provides receiver threshold configuration details.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				RNCK_THD				Reserved				RDT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Function
31-12 —	Reserved
11-8 RNCK_THD	Receiver NACK Threshold Value Specifies the number of consecutive NACKs transmitted by EMVSIM, for a given character, before the receive threshold error flag (RX_STATUS[RTE]) is triggered. A value of 0 indicates that RX_STATUS[RTE] is never 1. When EMVSIM receives a valid character, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. If CTRL[ANACK] = 0, then this field has no effect. 0000 - Zero threshold: RTE is not 1 0001 - RTE is triggered after 1 NACK is transmitted 0010 - RTE is triggered after 2 NACKs are transmitted 0011 - RTE is triggered after 3 NACKs are transmitted ... 1111 - RTE is triggered after 15 NACKs are transmitted
7-4 —	Reserved
3-0 RDT	Receiver Data Threshold Value Determines the number of bytes that must exist in the receive FIFO to trigger the receive data threshold interrupt flag (RX_STATUS[RDTF]). This value must not exceed the total number of bytes that can be stored in the receive FIFO.

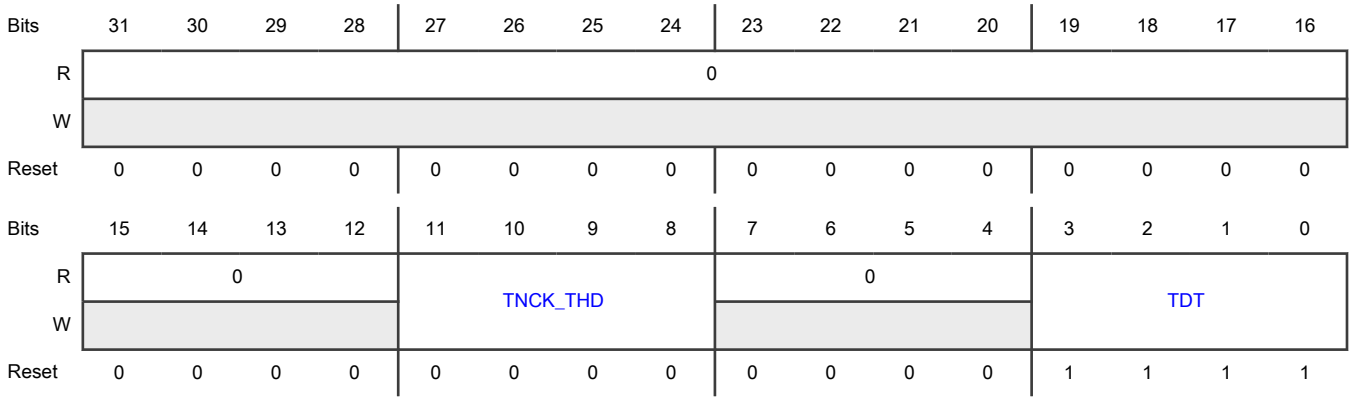
69.7.1.9 Transmitter Threshold (TX_THD)

Offset

Register	Offset
TX_THD	1Ch

Function
Provides transmitter threshold configuration details.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-8 TNCK_THD	Transmitter NACK Threshold Value Sets the NACK threshold for the transmitter. After the threshold number set by TNCK_THD has been reached for the current byte being transmitted, the error flag (TX_STATUS[TNTE] is set. Writing 1 to TX_STATUS[TNTE] causes the remaining transmissions queued in the transmit FIFO to be canceled, and no more transmissions can occur until software clears TX_STATUS[TNTE]. To trigger TX_STATUS[TNTE], a given byte being transmitted must reach the TNCK_THD threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next. 0000 - Error flag is never set; retransmission after NACK reception is disabled 0001 - Error flag is set after 1 NACK is received; 0 retransmissions occur 0010 - Error flag is set after 2 NACKs are received; at most 1 retransmission occurs 0011 - Error flag is set after 3 NACKs are received; at most 2 retransmissions occur ... 1111 - Error flag is set after 15 NACKs are received; at most 14 retransmissions occur
7-4 —	Reserved
3-0 TDT	Transmitter Data Threshold Value Sets the threshold value for the transmit FIFO at which TX_STATUS[TDTF] is set. When the number of bytes in the transmit FIFO is less than or equal to TDT, EMVSIM writes 1 to TX_STATUS[TDTF].

69.7.1.10 Receive Status (RX_STATUS)

Offset

Register	Offset
RX_STATUS	20h

Function
Provides receive status details.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				RX_CNT				0				RX_WPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		FEF	PEF	BGT_ERR	BWT_ERR	RTE	CWT_ERR	CRC_OK	LRC_OK	RDTF	RX_DATA	0			RFO
W			W1C	W1C	W1C	W1C	W1C	W1C				W1C				W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 RX_CNT	Receive FIFO Byte Count Indicates the number of bytes stored in the receive FIFO. 0000b - FIFO empty
23-20 —	Reserved
19-16 RX_WPTR	Receive FIFO Write Pointer Value Contains the value of the write pointer of the receive FIFO.
15-14 —	Reserved
13 FEF	Frame Error Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates if the current received byte did not have proper STOP bits. This does not cause a NACK transmission. This field does not cause an interrupt assertion. It is simply an indication that a framing error has occurred. 0b - No error 1b - Error
12 PEF	Parity Error Flag Indicates if a received byte had a parity error or not. A parity error causes transmission of NACK if CTRL[ANACK] = 1 and this field is not asserted. When ANACK is not = 1, PEF is asserted. This field also does not assert when CTRL[ICM] = 1; however, parity is checked in CTRL[ICM] = 1 mode. It is simply an indication that a parity error has occurred. 0b - No error 1b - Error
11 BGT_ERR	Block Guard Time Error Flag Indicates if the block guard time was too small or sufficient. The threshold is set by BGT_VAL[BGT]. 0b - Sufficient 1b - Too small
10 BWT_ERR	Block Wait Time Error Flag Indicates if the block wait time has been exceeded. The threshold is set by BWT_VAL[BWT]. 0b - Not exceeded 1b - Exceeded
9 RTE	Received NACK Threshold Error Flag Indicates whether the number of consecutive NACKs generated by EMVSIM, in response to receive parity errors for the byte being received, equals the value programmed in RX_THD[RNCK_THD]. This field is never set unless CTRL[ANACK] = 1. CTRL[SAPD] must be 1 to enable the threshold error to trigger the auto power-down sequence. Clearing this field also resets the internal counter for consecutive NACKs being transmitted for a given byte. If this field = 0, the number of NACKs generated by the receiver is less than the value programmed in RX_THD[RNCK_THD]. If this field = 1, the number of NACKs generated by the receiver is equal to the value programmed in RX_THD[RNCK_THD]. 0b - Less than 1b - Equal to
8 CWT_ERR	Character Wait Time Error Flag Indicates when the time between received characters is equal to or greater than the value programmed in CWT_VAL[CWT]. If this field = 0, no character wait time violation has occurred. If this field = 1, a character wait time violation has occurred. 0b - No CWT violation

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - CWT violation
7 CRC_OK	CRC Check OK Flag Indicates when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters and begins when CTRL[CRC_EN] becomes 1. The current CRC residual can be reset by three mechanisms: • Write 0 to CTRL[CRC_EN]. • Write 1 to CTRL[XMT_EN]. • Automatically by hardware when TX_STATUS[ETCF] becomes 1 at the end of a transmission. 0b - Current CRC value does not match remainder. 1b - Current calculated CRC value matches the expected result.
6 LRC_OK	LRC Check OK Flag Indicates when the calculated 8-bit LRC value is correct for the current input data stream. The value is calculated across all received characters and begins when CTRL[LRC_EN] becomes 1. You can reset the current LRC residual by three mechanisms: • Write 0 to CTRL[LRC_EN]. • Write 1 to CTRL[XMT_EN]. • Automatically by hardware when TX_STATUS[ETCF] becomes 1 at the end of a transmission. If this field is 0, then the current LRC value does not match the remainder. If this field is 1, then the current calculated LRC value matches the expected result (that is, zero). 0b - No match 1b - Match
5 RDTF	Receive Data Threshold Interrupt Flag Asserts an interrupt when total bytes in the receive FIFO are equal to or greater than the programmed receive threshold (RX_THD[RDT]). EMVSIM writes 1 to this field any time the number of unread bytes in the receive FIFO is equal to or greater than the value of the threshold. This flag can be cleared (= 0) by reading enough bytes out of the receive FIFO to bring the number of bytes remaining in the FIFO below the threshold. Another way to clear this flag is to set RX_THD[RDT] higher than the number of unread bytes currently in the FIFO. This flag creates an interrupt if INT_MASK[RDT_IM] = 0. If this flag = 0, the number of unread bytes in the receive FIFO is less than the threshold. If this flag = 1, the number of unread bytes in the receive FIFO is greater or than equal to the threshold. 0b - Less than threshold 1b - Greater than or equal to threshold
4 RX_DATA	Receive Data Interrupt Flag Asserts an interrupt when a new data byte is received and entered into the receive FIFO.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No new byte 1b - New byte
3-1 —	Reserved
0 RFO	Receive FIFO Overflow Flag Indicates that EMVSIM was unable to store received data due to already unread bytes in the FIFO (FIFO full or almost full). It does not necessarily indicate that data has been lost. If CTRL[ONACK] = 1, EMVSIM generates a NACK pulse on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the smart card, which implies that no data has actually been lost. In this case, this RFO flag is just an indicator that the situation has occurred, which may be helpful in system debug. For the case where ONACK is not set, a set RFO flag does indicate a loss of data, because all bytes received with this field = 1 are indeed lost (including the byte that caused RFO to be set). The RFO flag causes an interrupt if INT_MASK[RFO_IM] is enabled (RFO_IM=0). 0b - No overrun error 1b - Overrun error

69.7.1.11 Transmitter Status (TX_STATUS)

Offset

Register	Offset
TX_STATUS	24h

Function

Provides transmitter status details.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				TX_CNT				0				TX_RPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						GPCNT1_...	GPCNT0_...	TDTF	TFF	TCF	ETCF	TFE	0		TNTE
W							W1C	W1C		W1C	W1C	W1C	W1C			W1C
Reset	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 TX_CNT	Transmit FIFO Byte Count Indicates the number of bytes stored in the transmit FIFO. 0000b - FIFO empty
23-20 —	Reserved
19-16 TX_RPTR	Transmit FIFO Read Pointer Contains the value of the read pointer of the transmit FIFO.
15-10 —	Reserved
9 GPCNT1_TO	General Purpose Counter 1 Timeout Flag Indicates when the value in General Purpose Counter 1 equals the value in GPCNT1_VAL[GPCNT1]. If this field is 0, the GPCNT1 time has not been reached, or this field has been written to 0. If this field = 1, the general purpose counter has reached the GPCNT1 value. 0b - GPCNT1 not reached, or flag cleared 1b - GPCNT1 reached
8 GPCNT0_TO	General Purpose Counter 0 Timeout Flag Indicates when the value in General Purpose Counter 0 equals the value in GPCNT0_VAL[GPCNT0]. If this field is 0, the GPCNT0 time has not been reached, or this field has been written to 0. If this field = 1, the general purpose counter has reached the GPCNT0 value. 0b - GPCNT0 not reached, or flag cleared 1b - GPCNT0 reached
7 TDTF	Transmit Data Threshold Flag Indicates that the total number of bytes in the transmit FIFO is less than or equal to the programmed transmit data threshold (TX_THD[TDT]). EMVSIM writes 1 to this field any time the number of bytes in the transmit FIFO is less than or equal to the value set by TX_THD[TDT]. This flag can be cleared by writing enough bytes into the transmit FIFO to make the number of bytes in the FIFO greater than the threshold. Another way to clear this flag is to set the threshold lower than the number of bytes currently in the FIFO. This flag creates an interrupt if INT_MASK[TDT_IM] = 0. 0b - Threshold exceeded or this field written to 0 1b - Threshold not exceeded
6 TFF	Transmit FIFO Full Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates when the transmit FIFO has been written with the maximum number of bytes that it can store. Subsequent writes may be ignored. This flag can only be cleared by writing 1 to it or by writing 1 to CTRL[SW_RST]. 0b - Not full 1b - Full
5 TCF	Transmit Complete Flag Indicates whether the EMVSIM transmitter is ready for a new transmission. This flag becomes set when the guard time has expired after the last byte in the transmit FIFO (if CTRL[XMT_CRC_LRC] = 0) or the LRC or CRC byte (if CTRL[XMT_CRC_LRC] = 1) has been transmitted. This flag creates an interrupt if INT_MASK[TC_IM] = 1. 0b - Pending or incomplete 1b - Complete
4 ETCF	Early Transmit Complete Flag Indicates that the EMVSIM transmitter has finished sending the last byte in transmit FIFO (if CTRL[XMT_CRC_LRC] = 0) or finished sending the LRC or CRC byte (if CTRL[XMT_CRC_LRC] = 1). This field differs from TCF because it is set before the guard time of the last byte has elapsed. This flag creates an interrupt if INT_MASK[ETC_IM] = 0. 0b - Pending or incomplete 1b - Complete
3 TFE	Transmit FIFO Empty Flag Indicates that the EMVSIM transmit FIFO has been emptied. This field becomes 1 when the last byte in the transmit FIFO has been successfully transferred out of the EMVSIM transmitter to the smart card. It also becomes 1 when TNTE = 1. This flag creates an interrupt if INT_MASK[TFE_IM] = 0. 0b - Not empty 1b - Empty
2-1 —	Reserved
0 TNTE	Transmit NACK Threshold Error Flag Indicates that the transmit NACK threshold has been reached. When this field = 1, EMVSIM performs no further transmissions until this field = 0. Any data transmissions still pending in the transmit FIFO are aborted, and TCF, ETCF, and TFE flags are written to 1. This flag creates an interrupt if INT_MASK[TNACK_IM] = 0. 0b - Threshold not reached 1b - Threshold reached

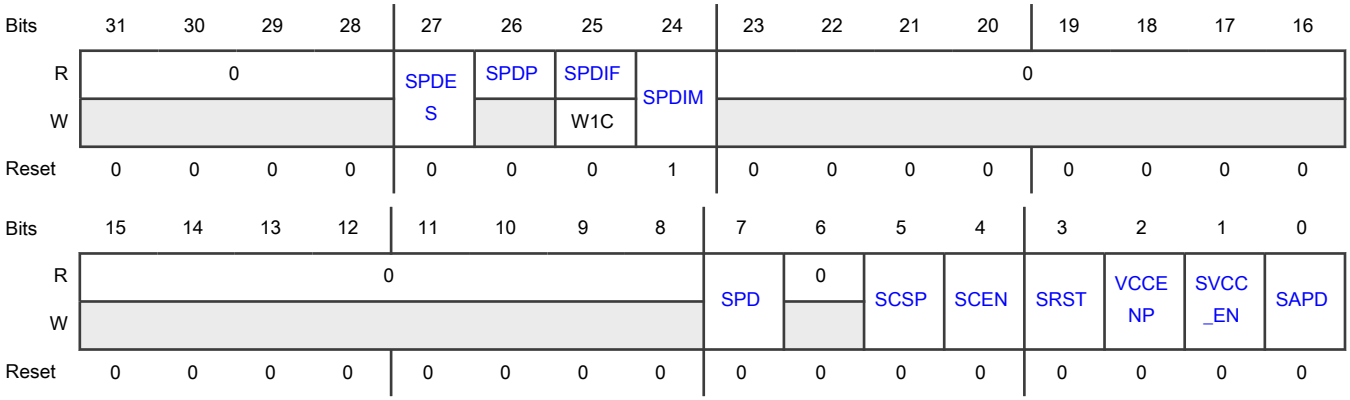
69.7.1.12 Port Control and Status (PCSR)

Offset

Register	Offset
PCSR	28h

Function
Provides control and status details related to the smart card port.

Diagram



Fields

Field	Function
31-28 —	Reserved
27 SPDES	SIM Presence Detect Edge Select Controls which edge of the Smart Card Presence Detect pin is used to detect the presence of the smart card. 0b - Falling edge 1b - Rising edge
26 SPDP	Smart Card Presence Detect Pin Status Reflects the state of the Smart Card Presence Detect pin. It is not a latched register bit, but instead a synchronized version of the state of the pin itself. 0b - Logic low 1b - Logic high
25 SPDIF	Smart Card Presence Detect Interrupt Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Contains the status flag to indicate that an insertion or removal of a smart card has been detected on the port. This field can create an interrupt to the MCU if SPDIM is 0, or initiate the auto power-down sequence if SAPD is 1. 0b - No insertion or removal 1b - Insertion or removal
24 SPDIM	Smart Card Presence Detect Interrupt Mask Enables the interrupt mask for the presence detect interrupt flag (SPDIF). If this field is 0, the SIM presence detect interrupt is enabled. If this field is 1, the SIM presence detect interrupt is masked. 0b - Enable 1b - Mask
23-8 —	Reserved
7 SPD	Auto Power-Down Control Starts the auto power-down sequence, if the auto power-down feature has been enabled by writing 1 to the SAPD field. This field automatically becomes 0 when power-down is complete. If this field is 1, either it starts auto power-down or else the power-down sequence is already in progress. 0b - No 1b - Yes
6 —	Reserved
5 SCSP	Smart Card Clock Stop Polarity Controls the polarity of the idle EMVSIM clock when the clock is disabled by the SCEN field. If this field is 0, the clock is logic 0 when stopped by SCEN. If this field is 1, the clock is logic 1 when stopped by SCEN. 0b - Logic 0 1b - Logic 1
4 SCEN	Clock Enable for Smart Card Enables the clock to the smart card. It can be forced low by hardware during the auto power-down sequence. 0b - Disable 1b - Enable
3 SRST	Reset Smart Card Controls the state of the reset line to the smart card. It can be forced low by hardware during the auto power-down sequence. Smart card reset signals are active low.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Assert 1b - Deassert
2 VCCENP	VCC Enable Polarity Control Controls the polarity of the SVEN output pad via the SVCC_EN field. When this field is 1, it inverts the value of the SVCC_EN field of this register and output to the SVEN output pad of this chip. If this field is 0, SVCC_EN is active high. The polarity of SVCC_EN is unchanged. If this field is 1, SVCC_EN is active low. The polarity of SVCC_EN is inverted. 0b - Active high 1b - Active low
1 SVCC_EN	Vcc Enable for Smart Card Controls the state of the SVEN pin on the smart card port. The SVEN pin controls the smart card Vcc enable in the power management device. It can be forced low by hardware during the auto power-down sequence. This field is XORed with the VCCENP field to control the polarity of the SVEN output pad. 0b - Disable 1b - Enable
0 SAPD	Auto Power Down Enable Enables the auto power-down feature. You must write 1 to this field before the auto power conditions are met. This field returns to 0 at the end of the auto power-down sequence. You can initiate the auto power-down function by: • Writing 1 to the SPD field by software • Writing 1 to RX_STATUS[RTE] • Writing 1 to SPDIF when the interrupt is used for detecting card removal You can still initiate a manual power-down through software by writing 1 to: • SCEN • SRST • VCC_EN This action will be effective no matter what the value of this field is. 0b - Disable 1b - Enable

69.7.1.13 Receive Data Read Buffer (RX_BUF)

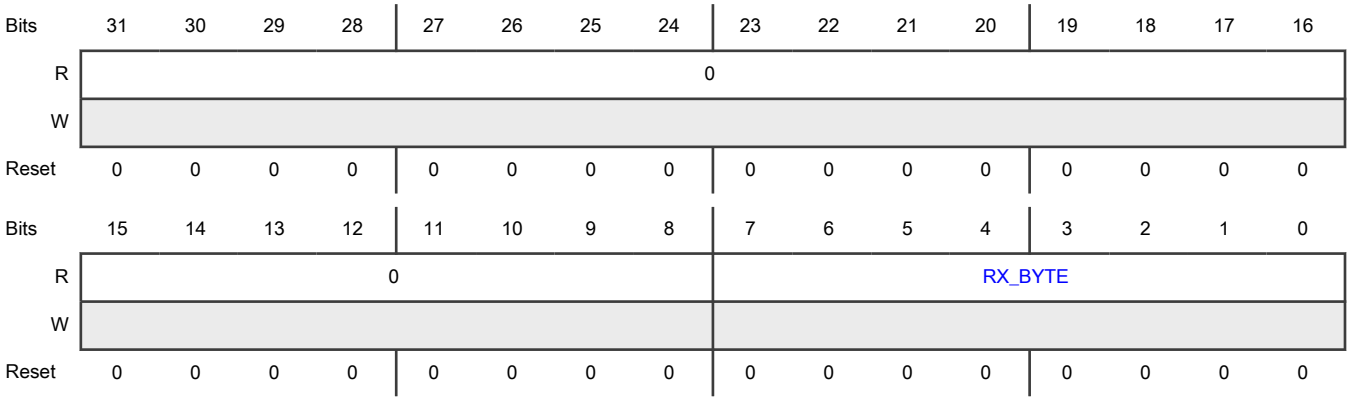
Offset

Register	Offset
RX_BUF	2Ch

Function

Provides the byte value from the top of the receive FIFO.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 RX_BYTE	Receive Data Byte Read Provides the byte value from the top of the receive FIFO. Each read access to this register increments the read pointer of the receive FIFO.

69.7.1.14 Transmit Data Buffer (TX_BUF)

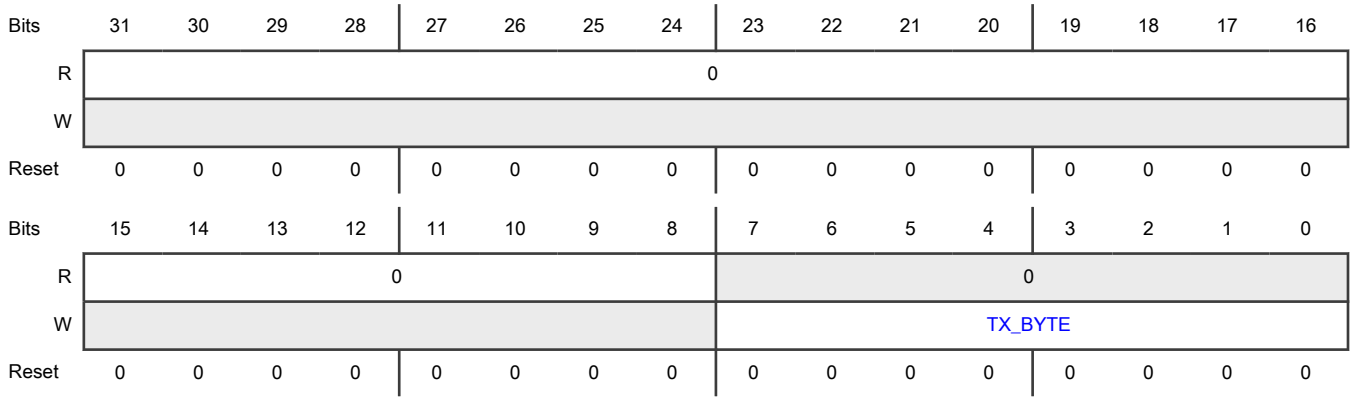
Offset

Register	Offset
TX_BUF	30h

Function

Fills the transmit FIFO with the bytes to be transmitted.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 TX_BYTE	Transmit Data Byte Fills the transmit FIFO with the bytes to be transmitted. The transmit FIFO can be written to only when the transmitter is enabled. Writing to this register while the transmitter is disabled leads to ignoring all write accesses to this register. Reads to this register return zeros. **NOTE** Writing more data to the transmit FIFO than it can hold causes a transmit FIFO full (TFF) error.

69.7.1.15 Transmitter Guard ETU Value (TX_GETU)

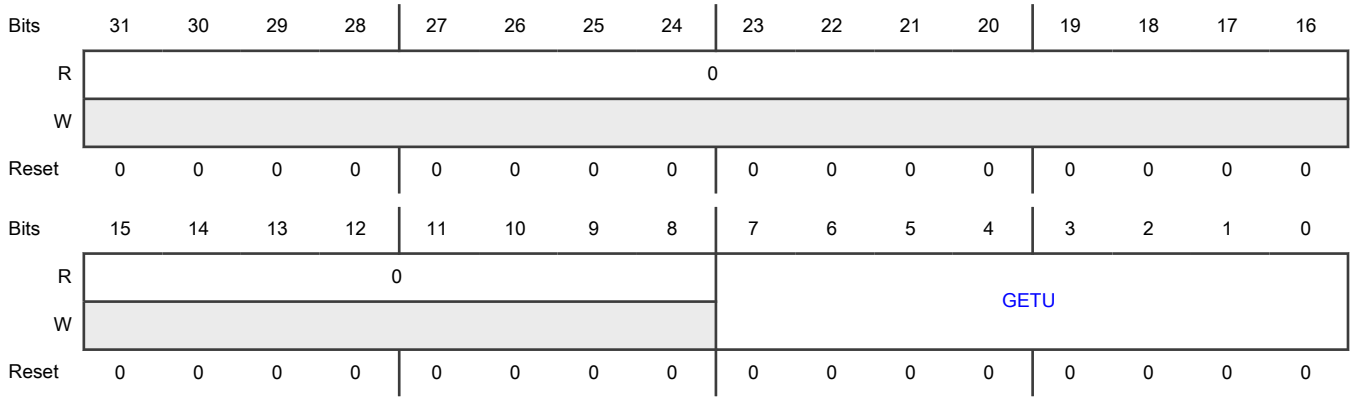
Offset

Register	Offset
TX_GETU	34h

Function

Controls the number of additional elementary time units (ETUs) inserted between bytes transmitted by the transmitter.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 GETU	Transmitter Guard Time Value in ETU Controls the number of additional elementary time units (ETUs) inserted between bytes transmitted by the EMVSIM transmitter. An ETU is equivalent to one bit time at the given baud rate (for example, the length of a start bit). The guard time has no effect on the EMVSIM receiver. A value of 00h inserts no additional ETUs, while a value of FEh inserts 254 additional ETUs. A value of FFh subtracts one ETU by reducing the number of stop bits from two to one. 00000000 - No additional ETUs inserted 00000001 - 1 additional ETU inserted ... 11111110 - 254 additional ETUs inserted 11111111 - Subtracts one ETU by reducing the number of stop bits from two to one

69.7.1.16 Character Wait Time Value (CWT_VAL)

Offset

Register	Offset
CWT_VAL	38h

Function

Specifies the number of ETU times allowed between characters.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CWT															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-16 —	Reserved
15-0 CWT	Character Wait Time Value Specifies the number of ETU times allowed between characters. The default value is FFFFh.

69.7.1.17 Block Wait Time Value (BWT_VAL)

Offset

Register	Offset
BWT_VAL	3Ch

Function

Specifies the maximum time between the time from the start bit of the last byte sent from EMVSIM to the start bit of the first byte sent from the smart card.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BWT															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWT															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-0 BWT	Block Wait Time Value Specifies the maximum block wait time value. The time from the start bit of the last byte sent from EMVSIM to the start bit of the first byte sent from the smart card must be less than the value in this register. If it is not, then EMVSIM writes 1 to the BWT_TO flag.

69.7.1.18 Block Guard Time Value (BGT_VAL)

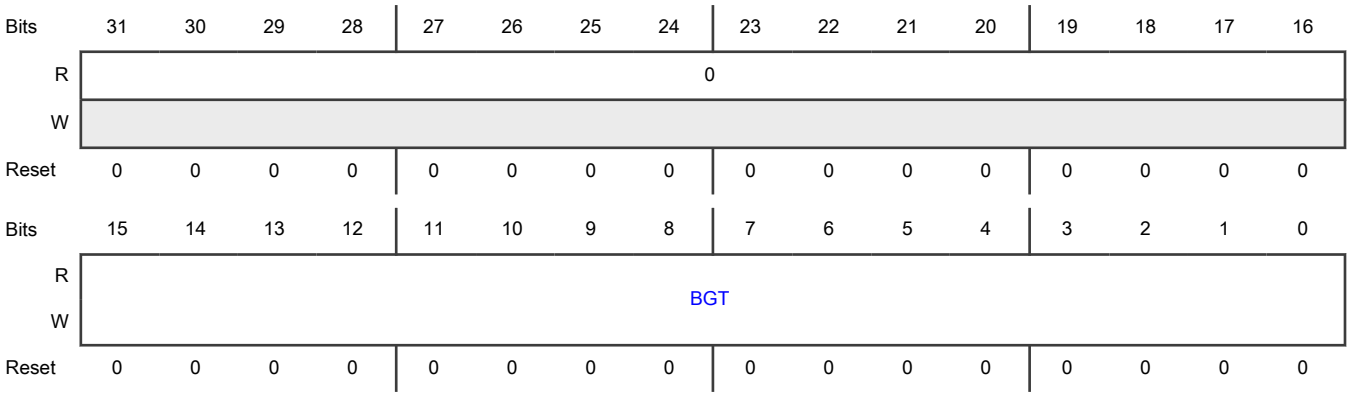
Offset

Register	Offset
BGT_VAL	40h

Function

Contains the block guard time. The time from the start bit of the last byte sent from EMVSIM to the start bit of the first byte sent from the smart card must be greater than this value.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 BGT	Block Guard Time Value Contains the block guard time value. The time from the start bit of the last byte sent from EMVSIM to the start bit of the first byte sent from the smart card must be greater than this value. If it is not, then EMVSIM writes 1 to the BGT flag.

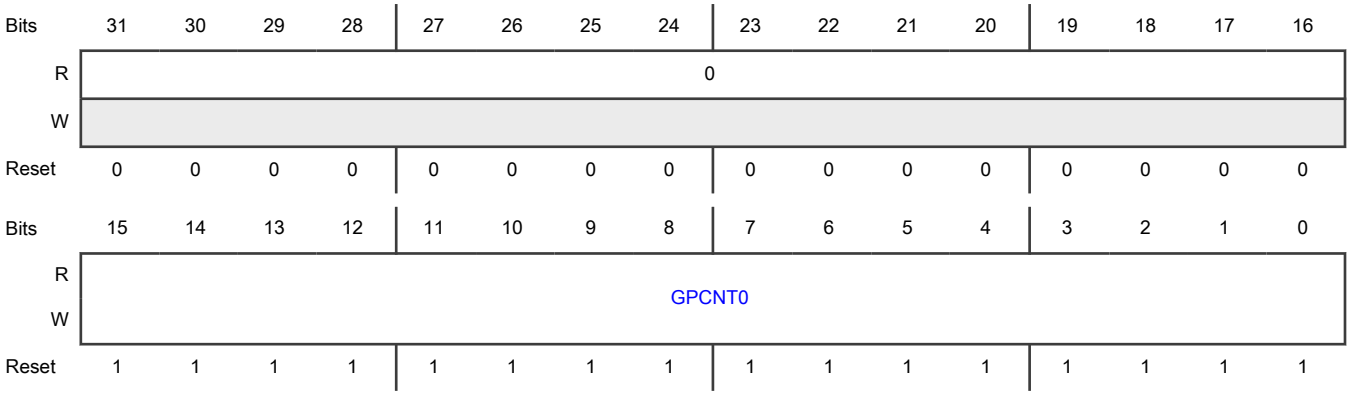
69.7.1.19 General Purpose Counter 0 Timeout Value (GPCNT0_VAL)

Offset

Register	Offset
GPCNT0_VAL	44h

Function
Contains a value to be compared to general purpose counter 0.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 GPCNT0	General Purpose Counter 0 Timeout Value Contains a value to be compared to general purpose counter 0. After the general purpose counter reaches this value, EMVSIM writes 1 to TX_STATUS[GPCNT0_TO]. This counter is intended for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration.

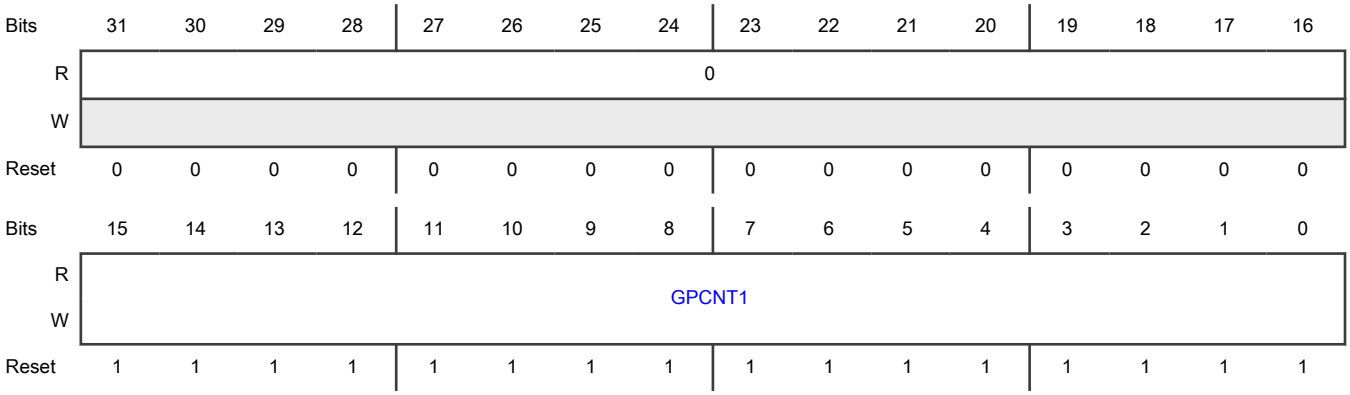
69.7.1.20 General Purpose Counter 1 Timeout Value (GPCNT1_VAL)

Offset

Register	Offset
GPCNT1_VAL	48h

Function
Contains a value to be compared to general purpose counter 1.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 GPCNT1	General Purpose Counter 1 Timeout Value Contains a value to be compared to general purpose counter 1. After the general purpose counter reaches this value, EMVSIM writes 1 to TX_STATUS[GPCNT1_TO]. This counter is intended for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration.

Chapter 70
Ethernet QoS (ENET)

70.1 Chip-specific ENET information

Table 637. Reference links to related information

Topic	Related module	Reference
Full description	ENET	Ethernet QoS
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

70.1.1 Module instances

This device has one instance of the Ethernet QoS module, ENET0.

70.1.2 Initialization

The Ethernet controller is configured as follows:

- Clock: In the SYSCON AHB Clock Control 2 (AHBCLKCTRL2) register, set ENET field, to enable the bus clock.
- Reset: In the SYSCON Peripheral Reset Control 2 (PRESETCTRL2) register, set ENET_RST field, to reset the Ethernet controller when needed.
- The Ethernet interrupts are connected to interrupt slots # 139, # 140, and # 141 in the NVIC. The Ethernet wake-up interrupt Ethernet wake-up packet indicator is connected to slot # 140 in the NVIC. See Nested Vectored Interrupt Controller (NVIC).
- Set the Ethernet mode to RMII or MII in the Ethernet PHY Interface Select (ENET_PHY_INTF_SEL) register in SYSCON.
- Set the sideband flow control for each channel.

70.2 Overview

NOTE

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

The Ethernet module enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard. The Ethernet interface contains a full-featured 10 Mbps or 100 Mbps Ethernet Media Access Controller (MAC) designed to provide optimized performance through the use of DMA hardware acceleration.

Features include control registers, full-duplex or half-duplex operation, flow control, control frames, hardware acceleration for transmit retry, receive packet filtering, and wake-up on LAN activity, supporting both wake-up and magic packet frames. Automatic frame transmission and reception with Scatter-Gather DMA off-load many operations from the CPU.

The Ethernet module is an AHB master connected to the AHB multilayer matrix and has access to internal SRAM and memory connected to the External Memory Controller for Ethernet data, control, and status information. A multi-layer AHB matrix connects the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters.

The Ethernet module interfaces with an off-chip Ethernet PHY using the Media Independent Interface (MII) or Reduced MII (RMII) protocol and with the on-chip Media Independent Interface Management (MIIM) serial bus.

70.2.1 Block Diagram

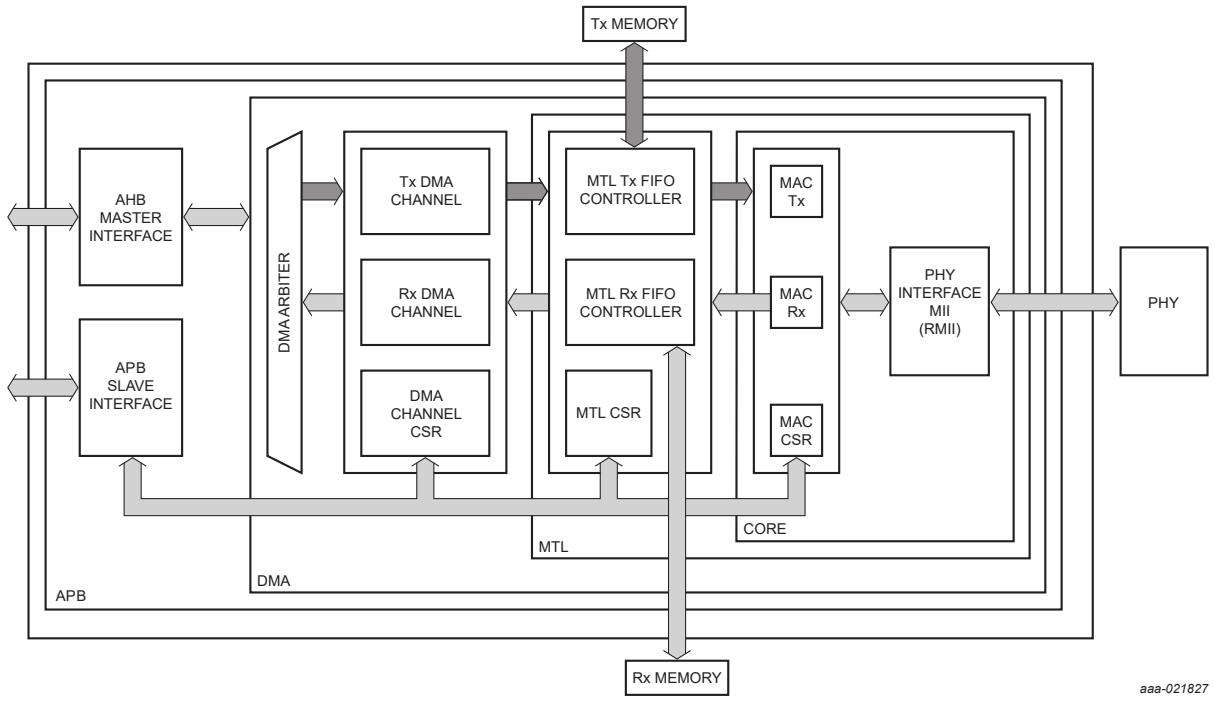


Figure 413. Ethernet block diagram

70.2.2 Features

key features of ENET module are:

- 10/100 Mbit/s
- Ethernet MAC IEEE 8023-2008
- DMA support
- IEEE 1588-2008 v2 timestamping driven by external PHY clock
- IEEE 1588 advanced timestamp support (IEEE 1588-2008 v2)
- Energy efficient Ethernet (IEEE 8023az-2010) for MII PHY
- Power management remote wake-up frame and magic packet detection
- Supports both MII and RMII (ver12) PHY
- Supports MDIO (Clause 22 and Clause 45) master interface to manage and configure PHY
- IP checksum offload support for TCP, UDP, or ICMP payloads encapsulated over IP
- Supports both full-duplex and half-duplex operation
  - Supports CSMA/CD Protocol for half-duplex operation
  - Supports IEEE 8023x pause packets and priority flow control for full-duplex operation
  - Optional forwarding of received pause control frames to the user application in full-duplex operation
  - Backpressure support for half-duplex operation
  - Automatic transmission of zero-quanta pause frame on de-assertion of flow control input in full-duplex operation

70.3 Functional description

NOTE

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

70.3.1 Power management and low power modes

This section describes the power management (PMT) mechanisms supported by the MAC. PMT supports the reception of network (remote) wake-up frames and magic packet frames. PMT does not perform the clock gate function, but generates interrupts for wake-up frames and magic packets received by the MAC. The PMT block sits on the receiver path of the MAC and is enabled with remote wake-up frame enable and magic packet enable. These enables are in the PMT control and status register and are programmed by the application.

When the power-down mode is enabled in the PMT, then all received frames are dropped by the Ethernet core and they are not forwarded to the application. The Ethernet comes out of the power-down mode only when either a magic packet or a remote wake-up frame is received and the corresponding detection is enabled.

70.3.1.1 Energy Efficient Ethernet

Energy Efficient Ethernet (EEE) is an operational mode that enables the IEEE 802.3 MAC sublayer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps. The Ethernet module supports the IEEE 802.3az-2010 for EEE. The LPI mode allows power saving by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionality to save power during periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY. The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

70.3.1.1.1 Transmit path functions

In the transmit path, the software must set the LPIEN bit of the MAC LPI control status register to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER(LST) field of MAC LPI timers control register. The PHY link status bit of the LPI control and status register indicates the link status of the PHY.

According to the Energy Efficient Ethernet standard (IEEE 802.3az-2010), the PHY must not stop the transmit clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts transmit enable (TX_EN).
2. Asserts transmit error (TX_ER). Sets TXD[3:0] to 0x1 The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.
3. Updates the status (TLPIEN bit of MAC LPI control status register and generates an interrupt. To bring the PHY out of the LPI state, that is, when the software resets the LPIEN bit, the MAC performs the following tasks:
4. Stops transmitting the LPI pattern and start transmitting the IDLE pattern
5. Starts the LPI TW TIMER
6. Updates the LPI exit status (TLPIEX bit of the MAC LPI control status register and generates an interrupt

The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the MAC LPI Timers control register.

70.3.1.1.2 Automated entry/exit of LPI mode in TX path

The MAC transmitter can be programmed to enter and exit LPI IDLE mode automatically based on whether it is IDLE for a specific period or has a packet to transfer. These modes are enabled and controlled by the MAC LPI control status register.

When the LPITXA (bit 19) and LPITXEN (bit 16) of the MAC LPI control status register are set, the MAC transmitter enters the LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI IDLE state and clears the LPITXEN bit as soon as any of the functions in the TX path (DMA, MTL, or MAC) becomes non-idle due to initiation of a packet transfer. In addition to the above, when bit 20 (LPIATE) is also set, the MAC transmitter enters LPI IDLE state only when the transmit path remains in an idle state (no activity) for the time indicated by the value in MAC LPI Entry Timer. In this mode also, the MAC transmitter exits the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPITXEN bit is not cleared but remains active so that reentry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both LPIATE and LPITXA bits are cleared, you can directly control the entry and exit of the LPI IDLE state by programming the LPITXEN bit.

70.3.1.1.3 Receive path functions

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[7:0] to 0x01.
3. The PHY de-asserts RX_DV. The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.
4. The MAC updates the RLPIEN bit of the MAC LPI control status register and immediately generates an interrupt.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the MAC LPI control status register and generates an interrupt immediately. The sideband signal (synchronous to Rx clock) is also asserted.

70.3.1.1.4 LPI timers

The transmitter maintains the following two timers that are loaded with the respective values from the MAC_LPI_Timers_Control register:

- LPI LS Timer
  - The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. You can enable the monitoring of the LUD bit of the MAC_PHYIF_Control_Status register by setting the PLSEN bit of the MAC_LPI_Control_Status register. The link status is indicated to the MAC by the LNKSTS bit of the MAC_PHYIF_Control_Status register or the value programmed by the software in the PLS bit of the MAC_LPI_Control_Status register. If the link status is not available in the MAC_PHYIF_Control_Status, the software should get the PHY link status by reading the PHY register and accordingly update the PLS bit. This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in bits 25:16 of the MAC_LPI_Timers_Control register. The GMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1,023 ms.
- LPI TW Timer
  - The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in bits 15:0 of MAC register 53 (LPI timers control register). The terminal count of the timer is the value of resolved transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count. The MAC supports the LPI TW TIMER in units of a microsecond. The LPI TW TIMER is 16-bits wide. Therefore, the software can program up to 65,535 µs.

70.3.1.1.5 LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The low-power interrupt is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the MAC LPI control status register.

70.3.1.2 Power management and wake-up

The power management (PMT) block supports the reception of magic packets as wake up packets, as well as the reception of network (remote) wake-up packets as wake up packets.

70.3.1.2.1 Magic packet detection

The magic packet frame is based on a method that uses Advanced Micro Device's magic packet technology to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

Only magic packets that are addressed to the device or a broadcast address are checked to determine whether they meet the wake-up requirements. Magic packets that pass the address filtering (unicast or broadcast)are checked to determine whether they meet the remote wake-on-LAN data format of 6 bytes of all ones followed by a MAC address appearing 16 times.

The application enables magic packet wake-up by writing a 1 to bit 1 of the PMT control and status register. The PMT block constantly monitors each frame addressed to the node for a specific magic packet pattern. Each frame received is checked for a 0xFFFF FFFF FFFF pattern following the destination and source address field. The PMT block then checks the frame for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the 0xFFFF FFFF FFFF pattern is scanned for again in the incoming frame. The 16 repetitions can be anywhere in the frame but must be preceded by the synchronization stream (0xFFFF FFFF FFFF). The device also accepts a multicast frame, as long as the 16 duplications of the MAC address are detected.

If the MAC address of a node is 0x0011 2233 4455, then the MAC scans for the data sequence:

- 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
- 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
- 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
- 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
- ...CRC

Magic packet detection is updated in the PMT control and status register for magic packet received. A PMT interrupt to the application triggers a read to the PMT CSR to determine whether a magic packet frame is received. When you enable the power-down mode in the PMT block, the MAC drops all received packets and does not forward any packet to the MTL Rx FIFO or the application. The MAC comes out of the power-down mode only when a magic packet is received and the corresponding detection is enabled.

70.3.1.2.2 Remote wake-up detection

The PMT block supports the reception of network (remote) wake-up packets as wake-up packets.

When the MAC is in sleep mode and the remote wake-up bit is enabled in PMT control and status register (0x00C0), normal operation is resumed after receiving a remote wake-up frame. The application writes all eight wake-up filter registers by performing a sequential write to address (0x00C4). The application enables remote wake-up by writing a 1 to bit 2 of the PMT control and status register.

PMT supports 16 programmable filters that allow support of different receive frame patterns. If the incoming frame passes the address filtering of filter command, and if filter CRC-16 matches the incoming examined pattern, then the wake-up frame is received.

Filter_offset (minimum value 12, which refers to the 13th byte of the frame) determines the offset from which the frame is to be examined. Filter byte mask determines which bytes of the frame must be examined. The 31st bit of the byte mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The wake-up frame is checked only for length error, FCS error, dribble bit error, MII error, collision, and to ensure that it is not a runt frame. Even if the wake-up frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. Wake-up frame detection is updated in the PMT control and status register for every remote wake-up frame received. A PMT interrupt to the application triggers a read to the PMT control and status register to determine the reception of a wake-up frame.

70.3.1.2.3 PMT interrupts

The PMT interrupt signal is asserted when a valid remote wake-up packet is received. When software resets the PWRDWN bit in the remote wake-Up packet detection register, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

When you enable the power-down mode in the PMT block, the MAC drops all received packets and does not forward any packet to the MTL Rx FIFO or the application. The MAC comes out of the power-down mode only when a remote wake-up packet is received and the corresponding detection is enabled. The PMT block is available in the receive path of MAC.

70.3.1.3 System considerations during power-down

MAC neither gates nor stops clocks when power-down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked with ENET_RX_CLK during power-down mode because it is involved in magic packet/wake-on-LAN frame detection. However, the transmit path and the application path clocks can be gated off during deep-sleep mode.

The PMT interrupt is asserted when a valid wake-up frame is received. This signal is generated in the receive clock domain.

The recommended power-down and wake-up sequences are as follows.

1. Disable the transmit DMA and wait for any previous frame transmissions to complete. These transmissions can be detected when transmit interrupt (see DMA channel status register bit TI) is received.
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the MAC configuration register.
3. Wait until the receive DMA empties all the frames from the Rx FIFO. This can be done by reading the appropriate bits of Debug registers in the DMA and MTL CSR space.
4. Enable power-down mode by appropriately configuring the PMT registers.
5. Enable the MAC receiver and enter power-down mode.
6. Gate the application and transmit clock inputs to the core (and other relevant clocks in the system) to reduce power and enter sleep mode.
7. On receiving a valid wake-up frame, the MAC asserts the PMT interrupt signal and exits the power-down mode.
8. On receiving the interrupt, the system must enable the application and transmit clock inputs to the core.
9. Read the PMT status register to clear the interrupt, then enable the other modules in the system and resume normal operation.

70.3.2 Flow control

This section describes the flow control for transmit and receive paths.

70.3.2.1 Transmit flow control

The transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end. To independently enable transmit flow control for each Rx Queue, set the TFE bit in the corresponding MAC queue Tx flow control register. The transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

70.3.2.1.1 Flow control in full-duplex mode

In full-duplex mode, one of the following packet types are used for flow control:

- IEEE 802.3x pause packets.
- Priority flow control (PFC) packets.

The PFC packets are used only when the enable data center bridging option is selected. The PFCE bit of MAC_Rx_Flow_Ctrl register determines whether PFC packets or IEEE 802.3x pause control packets are used for flow control. If the PFCE bit is set, the MAC sends the PFC packets.

describes the fields of a pause packet.

Table 638. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC	Control opcode Contains 0001 for IEEE 802.3x pause control packets; 0101 for PFC packets
PT	Contains pause time specified in the PT field of the MAC_Tx_FLOW_CTRL Q# register

When the FCB bit is set, the MAC generates and transmits a single pause packet. If the FCB bit is set again after the pause packet transmission is complete, the MAC sends another pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously transmitted pause packet, the application should program the pause time register with the appropriate value and then again set the FCB bit. Similarly, when the mti_flowctrl_i signal is asserted, the MAC generates and transmits a single pause packet. If the mti_flowctrl_i signal remains asserted at a configurable number of slot times before the pause time runs out, the MAC transmits a second pause packet. This process is repeated as long as the mti_flowctrl_i signal remains active. If the mti_flowctrl_i signal goes inactive prior to the sampling time, the MAC transmits a pause packet with zero-pause time (if the DZPQ bit in MAC_Q#_Tx_Flow_Ctrl register is set to 0) to indicate to the remote end that the receive buffer is ready to receive new data packets.

For PFC packets, you can specify the priority, pause time, and other controls for a queue in MAC_Q#_Tx_Flow_Ctrl register. The MAC supports independent flow control for each Rx Queue. There is one trigger input for each Rx Queue. Therefore, when multiple triggers come simultaneously, the MAC sends one PFC packet for each trigger. Based on the hardware trigger or software trigger through respective queues, the MAC sends a PFC packet with programmed pause time and priority. If multiple priorities are programmed in the same Rx Queue, multiple priorities are set for the PFC packet with the same pause time values. A separate pause timer is available for each Rx Queue.

70.3.2.1.2 Flow control in half-duplex mode

In half-duplex mode, the MAC uses the deferral mechanism for flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station back off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

The following table describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EFC bit of MTL_RxQ0_Operation_Mode register
- TFE bit of MAC_TX_FLOW_CTRL_Q0 register
- DM bit of the MAC configuration register flow control is similar for all queues.

Flow control is similar for all queues.

Table 639. Tx MAC flow control

EFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1.
1	1	0	The MAC transmitter performs backpressure when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx performs backpressure when Rx Queue level crosses the threshold set by bits 10:8 of MTL_RxQ0_Operation_Mode register.
0	1	1	The MAC transmitter sends the pause packet when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1.
1	1	1	The MAC transmitter sends the pause packet when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx sends a pause packet when Rx Queue level crosses the threshold set by bits 10:8 of MTL_RxQ0_Operation_Mode register.

70.3.2.2 Receive flow control

In the receive path, the flow control is functional only in the full-duplex mode. If any pause packet is received in the half-duplex mode, the packet is considered as a normal control packet. To enable the pause flow control Set the RFE bit in the MAC Rx flow control register.

70.3.3 Transmit and Receive FIFOs

The Transmit FIFO (Tx FIFO) buffers the data transferred from the host controller to the Ethernet module. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they can be transferred to the host controller. These are asynchronous FIFOs because they also transfer the data between the host controller clock and the MAC line clocks. Tx memory and Rx memory are required to be two-ported RAM of 35-bit width for 32-bit data bus widths, respectively. The extra bits are used for storing the byte-enable information.

When multiple queues are selected, all Tx Queue share the Tx FIFO memory and all Rx Queue share the Rx FIFO memory. The application can program the size of the FIFO memory allocated to each Tx or Rx Queue.

70.3.4 Multiple channels and queues support

The Ethernet module supports up to 2 queues and channels on Tx and Rx paths. The number of Tx channels are equal to the number of Tx Queues.

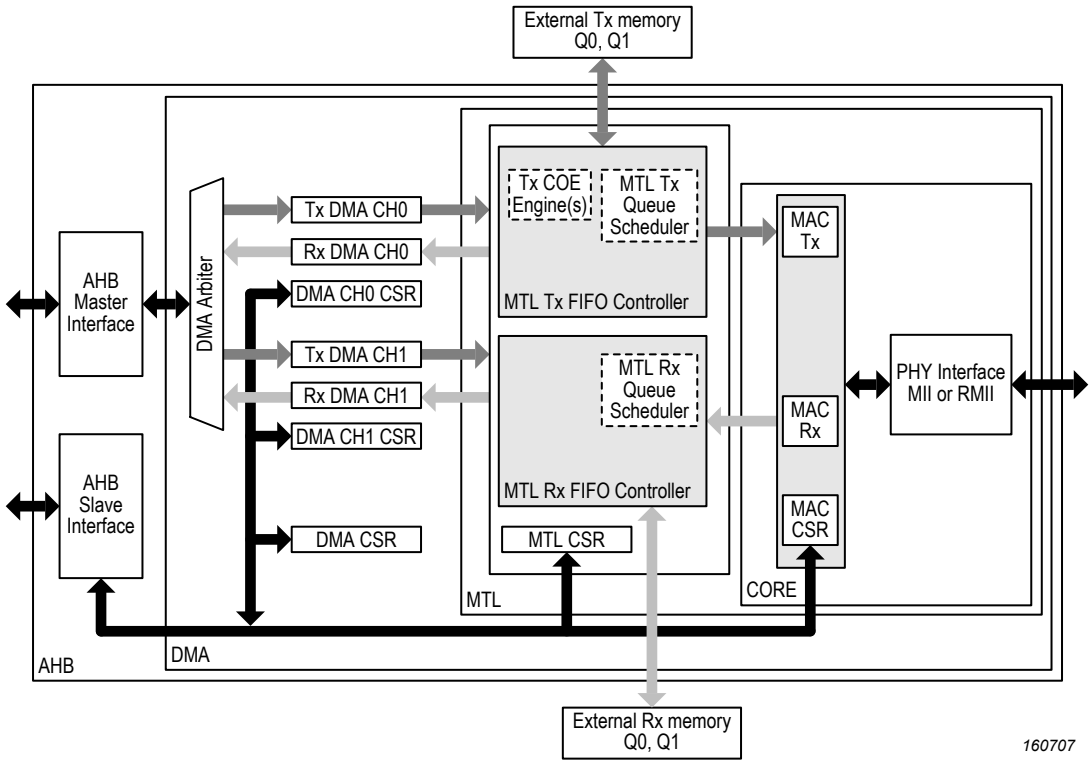


Figure 414. Multiple channels and queues

70.3.4.1 Support in the transmit path

Ethernet module supports up to two Tx Queues. The fixed priority scheme is the default priority scheme for the DMA channels. In a fixed priority scheme, the channel with the highest priority always wins the arbitration when it requests the bus. Channel 0 has a lower priority than channel 1. In weighted strict priority (WSP), the weight corresponds to the number of burst transfers given to a channel in one arbitration cycle. The unused burst transfers of one or more channels are reallocated based on the priority. The channel with the highest priority gets the unused burst transfer time before it is allocated to a channel with the next highest priority.

In weighted round-robin (WRR), all channels are serviced in round-robin order according to the weights settings. The TCW field of the corresponding DMA channel transmit control register provides the weight for each transmit channel. The configured weights correspond to the number of burst transfers given to a channel in one arbitration cycle. The unused or excess burst transfers are distributed equally to all channels.

70.3.4.2 Support in the receive path

In the Rx direction, the MTL Rx controller selects the Rx DMA for which it is transferring or reading the data from the Rx FIFO memory. This scheduling is based on the programming done in the respective MTL RxQ control register.

Each Rx DMA indicates when it is ready to transfer data and the size of the burst length (number of beats) that it has to transfer. The scheduler checks whether sufficient data (of requested burst length) is available to be transferred to these DMAs and then selects the Rx DMA that gets serviced using the programmed priorities.

70.3.4.3 Priority scheme for Tx DMA and Rx DMA

In the DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channels for accessing descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the DMA mode register specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a channel.

If the Tx DMA and Rx DMA of a channel are enabled, the user can specify which DMA gets the bus when the channel gets control of the bus. User can set the priority between the corresponding Tx DMA and Rx DMA by using the TXPR field of the DMA

mode register. For round-robin arbitration, the user can use the PR field of the DMA mode register to specify the weighted priority between the Tx DMA and Rx DMA.

Table 640. Priority Scheme for Tx DMA and Rx DMA

Bit 14	Bit 13	Bit 12	Bit11	Bit 1	Priority Scheme
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.
0	0	1	0	0	Rx has priority over Tx in ratio 2:1
0	1	0	0	0	Tx has priority over Rx in ratio 3:1
0	1	1	0	0	Tx has priority over Rx in ratio 4:1
1	0	0	0	0	Tx has priority over Rx in ratio 5:1
1	0	1	0	0	Tx has priority over Rx in ratio 6:1
1	1	0	0	0	Tx has priority over Rx in ratio 7:1
1	1	1	0	0	Tx has priority over Rx in ratio 8:1
x	x	x	1	1	Tx always has priority over Rx
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.
0	0	1	1	0	Tx has priority over Rx in ratio 2:1
0	1	0	1	0	Tx has priority over Rx in ratio 3:1
0	1	1	1	0	Tx has priority over Rx in ratio 4:1
1	0	0	1	0	Tx has priority over Rx in ratio 5:1
1	0	1	1	0	Tx has priority over Rx in ratio 6:1
1	1	0	1	0	Tx has priority over Rx in ratio 7:1
1	1	1	1	0	Tx has priority over Rx in ratio 8:1

Bits 14, 13, 12, 11, and 1 are not valid if one of the paths (transmit or receive) is not present in a channel. The transmit and receive paths are always present in channel 0.

70.3.4.4 Multiple queues and channels support in Ethernet MTL

The Ethernet module supports up to two Tx and Rx Queues. You can control a Tx Queue through the corresponding MTL TxQ operation mode register. Also, you can control an Rx Queue through the corresponding MTL RxQ operation mode register.

70.3.4.5 Rx Queue to DMA mapping

The packets in the MTL Rx Queues can be routed to any one of the two DMA channels by programming the MTL RxQ DMA map registers for queues 0, 1. The following types of Rx Queue to DMA mapping is possible through programming:

1. Static mapping

2. Dynamic (per packet) mapping

70.3.4.5.1 Static mapping

In this mode, all the packets of an Rx Queue are connected to a specific DMA channel. For example, all the packets from Rx Queue 0 can be routed to a DMA channel by programming Q0MDMACH (bits 3:0) and Q0DDMACH (bit 7 = 0) of the MTL RxQ DMA Map0 register. Similarly, packets from other Rx Queues can be routed to any DMA channel by programming register fields corresponding to each queue.

70.3.4.5.2 Dynamic (per packet) mapping

In this mode, the destination DMA channel of a packet that is read from an Rx Queue is not constant but decided independently for each packet. For example, if you set the Q1DDMACH bit of the MTL RxQ DMA map register, the static mapping is disabled for Rx Queue 1 and the value in Q1MDMACH is ignored. The destination DMA channel is decided by the MAC Core receiver for each packet, based on Ethernet DA-based DMA Selection. The DA address of the received packet is compared against the programmed DA values in MAC address registers. If the address matches any of the programmed values, the corresponding DCS field (when enabled) determines the destination DMA channel number. If the above operations is unable to make a successful match/decision, then the packet is routed to DMA channel 0 by default.

70.3.4.6 Rx side routing from MAC to queues

The MAC routes the Rx packets to the Rx Queues based on the following packet types:

1. AV control packets
2. Untagged IEEE 1588 PTP over Ethernet packets

The AV control Rx packets can be routed based on the Rx Queue number specified in the AVCPQ field in the MAC RxQ Ctrl1 register, and the corresponding Rx Queue is enabled for AV through the corresponding RXQEN field in MAC RxQ Ctrl0 register.

The untagged IEEE 1588 PTP over Ethernet Rx packets can be routed based on the Rx Queue number specified in the AVPTPQ field in MAC RxQ Ctrl1 register and the corresponding Rx Queue is enabled for AV through the corresponding RXQEN field in the MAC RxQ Ctrl0 register. This type of Rx packet routing is available when the AV feature and multiple Rx Queues are selected in the configuration.

If the Rx packet cannot be classified in any of the defined packet types for routing, it will be routed through Rx Queue 0.

70.3.4.7 Rx side arbitration between DMA and MTL

After completing the current packet processing, the DMA channel controller fetches the next descriptor. After the descriptor fetching is complete, the DMA channel controller evaluates the amount of data to be transferred to the Rx buffer based on the programmed PBL and Rx buffer length. Accordingly, it requests the MTL to transfer the data. After servicing the current request, the MTL Rx Queue arbitration scheme selects Rx Queue based on the arbitration scheme (RAA field of the MTL operation mode register) and the weights programmed in the corresponding queue receive control register. The arbitration is done among queues for which DMA is ready to serve. After the Rx Queue is selected, the PBL (Programmable Burst Length) amount of data is read out from that queue and is routed to the Rx DMA channel based on the Rx channel selection criteria.

The arbitration takes place when every PBL data transfer is completed and descriptors are ready for processing from at least one DMA channel.

70.3.4.8 Tx side arbitration between DMA and MTL

Because the number of Tx DMA channels is always equal to the number of Tx Queues in the MTL, they are always mapped directly. For instance, each Tx DMA pushes data into its respective Tx Queue assigned to it. The main reason for this strategy is to optimize and increase the efficiency of the DMA data transfer, as well as to simplify the TX checksum engines in the MTL.

The data inside each Tx Queue is stored in packets. Hence, if two DMAs are allowed to transfer data into the same queue when a Tx DM A starts a packet transfer, the other DMA cannot transfer data unless the previous packet is completely pushed-in. This means that the second DMA remains idle until the first packet is transferred. Hence, each DMA is always connected directly to its corresponding Tx Queue.

70.3.4.9 Audio Video Bridging

The Ethernet module supports the AV data transfer in 100 Mbps modes. The AV feature enables the transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

1. IEEE 802.1Qav-2009: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio-video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.
2. IEEE 802.1Qat-2009: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network.
3. IEEE 802.1AS-2011: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

70.3.4.9.1 Transmit path functions

When the AV feature is selected, the transmit paths of Queue 0 and the highest enabled queue are enabled by default. The transmit path of Queue 0 supports the strict-priority algorithm, and it is used for best-effort traffic. For a queue, the strict-priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict-priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programmed threshold limit.

The transmit paths of additional queues support traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

1. The queue contains one or more packets.
2. The credit for the queue is positive as per the algorithm.

The credit-based shaper algorithm can be disabled for all queues or lower-priority queues. For example, you can disable the credit-based shaper algorithm for Queue 1 and Queue 2 or only for Queue 1. Do not disable the credit-based shaper algorithm for Queue 2 and enable it for Queue 1. When you disable the credit-based shaper algorithm for a queue, the channel uses the default strict-priority algorithm. Each transmit DMA has a separate descriptor chain for fetching the transmit data. The transmit channel that gets access to the system bus depends on the DMA arbiter.

The transmit path has a shared FIFO (MTL layer) for each queue, as shown in Multiple channels and queues support. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a queue, the corresponding queue is selected for transmission if the following conditions are true:

1. If the packet is available in the channel and has a positive or zero credit.
2. If the higher priority queue has no packet waiting in the FIFO.

If the credit-based shaper algorithm is disabled for all queues, the packet to be transmitted from a queue is selected based on the priority scheme.

70.3.4.9.2 Receive path functions

When the AV feature is selected, the receive path of Queue 0 is enabled by default. All traffic is received on this channel. You can enable the receive paths of additional queues. By enabling the receive paths of multiple channels, you can demultiplex the received data to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if it is an AV packet and its corresponding VLAN Priority tag value. This status is updated in the extended status field of the receive descriptor as explained in "receive descriptor". All received packets with the Ether Type field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

1. AV data packets: The AV data packets are always tagged. The tagged AV data packets are received based on the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program bits 15:8 in the transmit flow control register of the corresponding queue.
2. Similar to the AV data packets, the tagged AV control packets are received based on the programmed priority value.

In addition to the AV packets, can be received the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on Queue 0. To receive these packets on any other queue, you need to program bits 6:4 of the MAC Rx Queue control1 register (MAC_RXQ_CTRL1).

70.3.4.9.3 Credit based shaper algorithm

The MTL queue scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in Queue 0. You can program the additional queues to use the credit-based shaper algorithm.The following sections provide information about how you can implement the credit-based shaper algorithm:

1. Credit value
2. idleSlopeCredit and sendSlopeCredit Values
3. Bandwidth status

70.3.4.9.3.1 Credit value

The credit value is accumulated every transmit clock cycle, that is, 40 ns for 100 Mbps. The credit to be added or subtracted per cycle can be fractional based on the required idleSlope and sendSlope values.

Table 641. Credit value Per transmit cycle example

Mode	Values	Description
100 Mbps	portTransmitRate = 100 Mbps	
	idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class)	credit = 2.8 bits accumulates per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted.
	sendSlope = 30 Mbps	credit = 1.2 bits drains per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO based on the slot number in the transmit descriptor (if enabled) or the bandwidth available on the AMBA AHB application bus.

The credit for a queue builds up only when the packet is available but it cannot be transmitted because the MAC is sending a packet from another queue. The Ethernet block supports another mode in which the credit can build up in advance for a queue in which no packet is available in the respective part of the FIFO. This enables sending a burst of high-priority traffic in a queue as soon as the data is available. You can enable this mode with bit 1 of the CBS control registers of corresponding queues.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

70.3.4.9.3.2 idleSlopeCredit and sendSlopeCredit Values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1,024, such as, 2.8 x 1,024 = 2,867 and 1.2 x 1,024 = 1,229. In addition, the software must program the hiCredit and loCredit values, scaled by 1,024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the

values to be programmed in the hiCredit and loCredit registers of the corresponding channel are 12,000 x 1,024 bits and two's complement of 3,036 x1,024, respectively.

70.3.4.9.3.3 Bandwidth status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register gives the average number of bits transmitted during the previously programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in bits 6:4 of the CBS control register of the respective queue. For example, if you have programmed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue. A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, Queue 1 the credit-based shaper algorithm (with reserved bandwidth of 50%) and Queue 0 uses the strict-priority algorithm. If Queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by Queue 0.

70.3.4.9.4 Slot number function with Audio Video Bridging

In AV mode the slot number function to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals. Slot number can be programmed to a particular slot at which the DMA should fetch the data from system memory in the transmit descriptor Word 3 (TDES3). This 4-bit field allows the application to schedule data up to 16 slots of 125 µs each. This field is applicable only for the AV channels. When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated every 125 µs of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the second field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state till there is a match.

To enable the DMA to fetch the data only if it matches the current slot or the next two slots, you can program bit 1 of the slot function control and status register of the corresponding DMA channel. You can enable the check for slot number by setting bit 0 of the slot function control and status register of the corresponding DMA channel. When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, bits 19:16 indicate the value of the reference slot number in DMA.

If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

See AV feature.

70.3.4.10 Queue modes

You can enable a Tx Queue for generic, AV, or all types of traffic. When you select multiple Tx Queues, the queues are enabled for generic queuing using WRR or WSP algorithms. The queuing is based on the CBS or SP algorithms.You can enable an Rx Queue for generic, AV, or all types of traffic. A particular queue enabled for generic or AV based routing is determined by the RXQEN field of corresponding queue. When you select the enable audio video bridging option, AV is enabled for all selected Rx Queues.

The AV control packets (tagged or untagged) are routed based on the AVCPQ field of the MAC Rx Queue control 1 register (MAC_RXQ_CTRL1). The priority should be assigned in the following order:

1. AV queue (high priority)
2. Best-effort queue (low priority)

The software should put packets with correct priorities in the respective programmed queue on Tx side. The MAC uses the programmed priorities for blocking the Tx Queues when a PFC packet is received. If a single queue is selected for multiple priorities and PFC is enabled, the entire queue is paused if one or more priorities in the queue are paused.

70.3.5 TCP/IP offloading

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. Therefore, the MAC has a checksum offload engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path. By using the Checksum Offload features, the software can offload the task of checksum insertion and checking to the hardware. This feature helps in improving the throughput.

In the transmit path MAC calculates the checksum and inserts it in the Tx packet. In the receive path, MAC verifies the received checksum with the internally calculated checksum and provides the status.

70.3.5.1 Transmit checksum offload engine

The MAC has a checksum offload engine (COE) to support checksum calculation and insertion in the transmit path, using which, the software can offload the task of checksum insertion to the hardware. In the transmit path MAC calculates the checksum and inserts it in the Tx packet. This feature helps in improving the throughput. The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits 17:16).

The checksum for TCP, UDP, or ICMP is calculated over a complete packet and then inserted into its corresponding header field. It must be made sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter. The reason is that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid dead-lock. When reading starts, the COE fails and consequently, all succeeding packets may get corrupted because of improper recovery. Therefore, you must enable the checksum insertion only in the packets that are less than the following number of bytes in size:

- TXFIFO_SIZE – ((PBL + 6)*(DATAWIDTH/8))

The PBL is the programmed burst length in the DMA Bus mode register.

70.3.5.1.1 IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit header checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the type field of Ethernet packet has the value 0x0800 and the version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

70.3.5.1.2 TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP checksum engine processes the IPv4 or IPv6 header and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the checksum field of the input packet. This engine includes the checksum field in the checksum calculation and then replaces the checksum field with the final calculated checksum.
- The engine ignores the checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation and overwrites the checksum field with the final calculated value.

The result of this operation is indicated by the payload checksum error status bit in the transmit status vector (bit 12 in). This engine sets the payload checksum error status bit when it detects that the packet is forwarded to the MAC transmitter engine in the store and forward mode without the end of packet (EOP) is written to the FIFO, or when the packet ends before the number of bytes indicated by the payload length field in the IP header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 642. Transmit checksum offload engine functions for different frame types

Frame type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 frame	No	No
IPv4 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K frames is enabled in MAC) but less than or equal to the frame size constraint specified in	Yes	Yes
IPv6 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K frames is enabled in MAC) but less than or equal to the frame size constraint specified in	N/A	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 frame has IP options (IP header is longer than 20 bytes)	Yes	Yes
Frame is an IPv4 fragment	Yes	No
IPv6 frame with the following next header fields in main or extension headers:	N/A	
• Hop-by-hop options (in IPv6 main header)		Yes
• Hop-by-hop options (in IPv6 extension header)		No
• Destinations options		Yes
• Routing (with segment left 0)		No
• Routing (with segment left > 0)		No
• TCP, UDP, or ICMP		Yes
• Authentication		Yes
• Any other next header field in main or extension headers		No
IPv4 frame has TCP header with Options fields.	Yes	Yes
IPv4 Tunnels: • IPv4 frame in an IPv4 tunnel • IPv6 frame in an IPv4 tunnel	• Yes (IPv4 tunnel header) • Yes (IPv4 tunnel header)	• No • No
IPv4 frame has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Yes	Yes
IPv6 frame has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	N/A	Yes

Table continues on the next page...

Table 642. Transmit checksum offload engine functions for different frame types (continued)

Frame type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	N/A	No

70.3.5.2 Receive checksum offload engine

The MAC has a checksum offload engine (COE) to support packet error detection in the receive path. In the receive path, MAC verifies the received checksum with the internally calculated checksum and provides the status. Here, both IPv4 and IPv6 packets in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for values 0x0800 or 0x86DD respectively in the Type field of the received Ethernet packet.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP header error bit is set for any mismatch between the indicated payload type (Ethernet type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header). This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a payload checksum error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 643. Receive checksum offload engine functions for different frame types

Frame type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6	No	No
IPv4 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)	Yes	Yes
IPv6 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K frames is enabled in the MAC)	N/A	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 frame has IP options (IP header is longer than 20 bytes)	Yes	Yes
Frame is an IPv4 fragment	Yes	No
IPv6 frame with the following next header fields in main or extension headers:	N/A	
Hop-by-hop options (in IPv6 main header)		Yes

Table continues on the next page...

Table 643. Receive checksum offload engine functions for different frame types (continued)

Frame type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Hop-by-hop options (in IPv6 extension header)		No
Destinations options		Yes
Routing (with segment left 0)		Yes
Routing (with segment left > 0)		No
TCP, UDP, or ICMP		Yes
Any other next header field in main or extension headers		No
IPv4 frame has TCP header with Options fields.	Yes	Yes
IPv4 Tunnels: • IPv4 frame in an IPv4 tunnel • IPv6 frame in an IPv4 tunnel	• Yes (IPv4 tunnel header) • Yes (IPv4 tunnel header)	• No • No
IPv6 Tunnels: • IPv4 frame in an IPv6 tunnel • IPv6 frame in an IPv6 tunnel	N/A	• No • No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled)	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled)	N/A	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	N/A	No

70.3.6 Loopback mode

The MAC supports loopback of transmitted packets to its receiver. To enable this feature, program the LM bit of the MAC configuration register. Loopback mode can be enabled for all PHY interfaces. The data is always looped back on the MII interface irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding interface block.

70.3.6.1 Guidelines for using loopback mode

The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (CRS) or collision (col) signal inputs get sampled which may result in issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip (for example, in FPGA setup), the Tx and Rx clocks should be externally generated and should provide these clocks to the MAC.
- Do not loop back big packets. Big packets may get corrupted in the loopback FIFO.

The transmit and receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the PHY transmit data to the receive path.

The depth of FIFO is nine. The FIFO is free-running to write on the write clock and read on every read clock.

At the start of each packet readout of the FIFO, the write and read pointers get re-initialized to have an offset of 4. This avoids overflow or underflow during a packet transfer. This also ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO. At the end of every received packet, the receive protocol engine module generates received packet status and sends it to the receive packet controller module. The control, missed packet, and filter fail status are added to the receive status in the receive packet controller module. The MAC does not process ARP or PMT packets that are looped back.

70.3.7 PHY interfaces

70.3.7.1 Station management agent

The host controller can access the PHY registers through the Station Management Agent (SMA) module. SMA is a two-wire station management interface (MIM).

SMA module supports accessing up to 32 PHYs. The host controller can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module.

70.3.8 IEEE 1588 timestamps

The IEEE 1588 standard defines a protocol, Precision Time Protocol (PTP), that enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The Ethernet module supports both the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2) standards. The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588-2008 supports PTP transported over Ethernet. The Ethernet provides programmable support for both standards. It supports the following features:

- Supports both timestamp formats.
- Provides an option to take a snapshot of all packets or only PTP type packets.
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent.
- Provides an option to select the node to be a master or slave for ordinary and boundary clocks.
- Identifies the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status.
- Provides an option to measure sub-second time in digital or binary format.

70.3.8.1 Clock types

The Ethernet controller supports the following clock types defined in the IEEE 1588-2008 standard:

- Ordinary clock
- Boundary clock
- End-to-end transparent clock
- Peer-to-peer transparent clock

70.3.8.1.1 Ordinary clock

The ordinary clock in a domain supports a single copy of the protocol. The ordinary clock has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecommunications applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. The ordinary clock supports the following features:

- Maintains the data sets such as timestamp values.

Table 644. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

For an ordinary clock, you can take the snapshot of either one of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the control bit (TSVER2ENA) and selecting the snapshot mode in MAC_TIMESTAMP_CONTROL.

70.3.8.1.2 Boundary clock

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy, and signaling terminate in the protocol engine of the boundary clock and are not forwarded. The PTP message type status given by the core helps you to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- The clock data sets are common to all ports of the boundary clock.
- The local clock is common to all ports of the boundary clock.

Therefore, the features of the ordinary clock are also applicable to the boundary clock.

70.3.8.1.3 End-to-end transparent clock

The end-to-end transparent clock supports the end-to-end delay measurement mechanism between slave clocks and the master clock. The end-to-end transparent clock forwards all messages like a normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the ingress port to the egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet inside the end-to-end transparent clock is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both ingress and egress ports only for the messages mentioned in the table below.

You can take the snapshot by setting the snapshot select bits (SNAPTYPSEL) to 10 in MAC_TIMESTAMP_CONTROL.

Table 645. End-to-end transparent clock: PTP messages for which a snapshot is taken for transparent clock implementation

SYNC
FOLLOW_UP

70.3.8.1.4 Peer-to-peer transparent clock support

In this type of clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer. Hence support for taking a snapshot for the event messages related to Pdelay is added.

Table 646. End-to-end transparent clock: PTP messages for which a snapshot is taken for transparent clock implementation

SYNC
Pdelay_Req
Pdelay_Resp

The transparent clock corrects only the sync and follow-up message. As discussed earlier this can be achieved using the message status provided.

The type of clock to be implemented is configurable through the control register. To ensure that the snapshot is taken only for the messages indicated in the table for the corresponding clock type, the TSEVNTENA (enable timestamp snapshot for event messages) bit has to be set.

70.3.8.2 Delay request-response mechanism

The system or network is classified into master and slave nodes for distributing the timing/clock information. The following figure shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

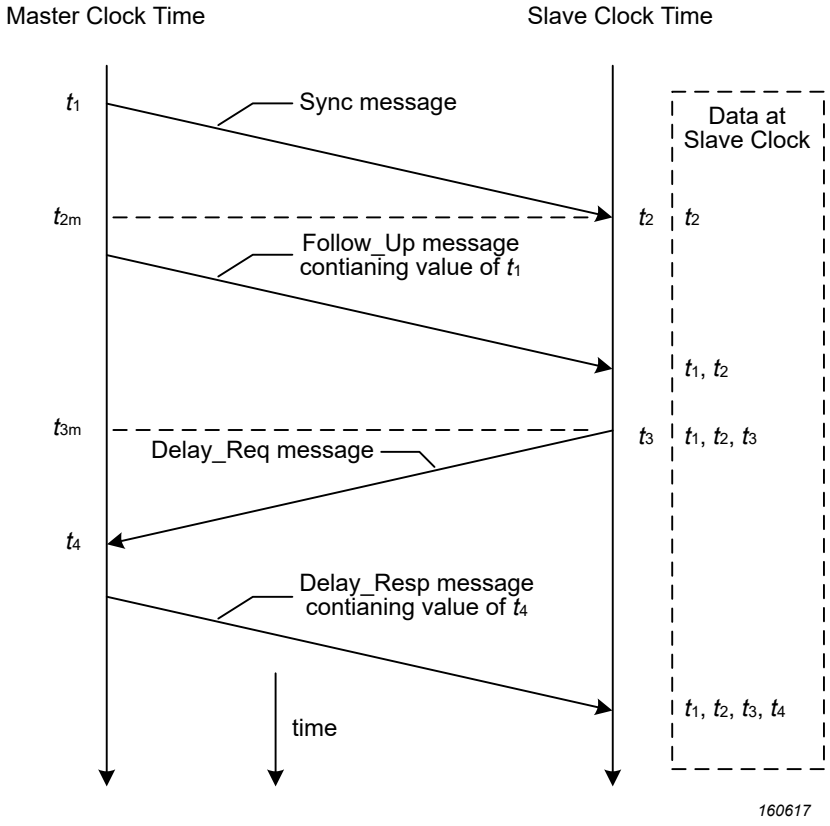


Figure 415. Networked time synchronization

The PTP uses the following process:

1. The master broadcasts the PTP sync messages to all its nodes. The sync message contains the master.s reference time information. The time at which this message leaves the master.s system is t1. This time must be captured, for Ethernet ports, at MII.
2. The slave receives the sync message and also captures the exact time, t2, using its timing reference.

3. The master sends a Follow_up message to the slave, which contains t1 information for later use.
4. The slave sends a Delay_Req message to the master, noting the exact time, t3, at which this frame leaves the MII.
5. The master receives the message, capturing the exact time, t4, at which it enters its system.
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the UDP layer. However, hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

70.3.8.3 Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 version supports peer-to-peer PTP (Pdelay) message in addition to sync, delay request, follow-up, and delay response messages. The below figure shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

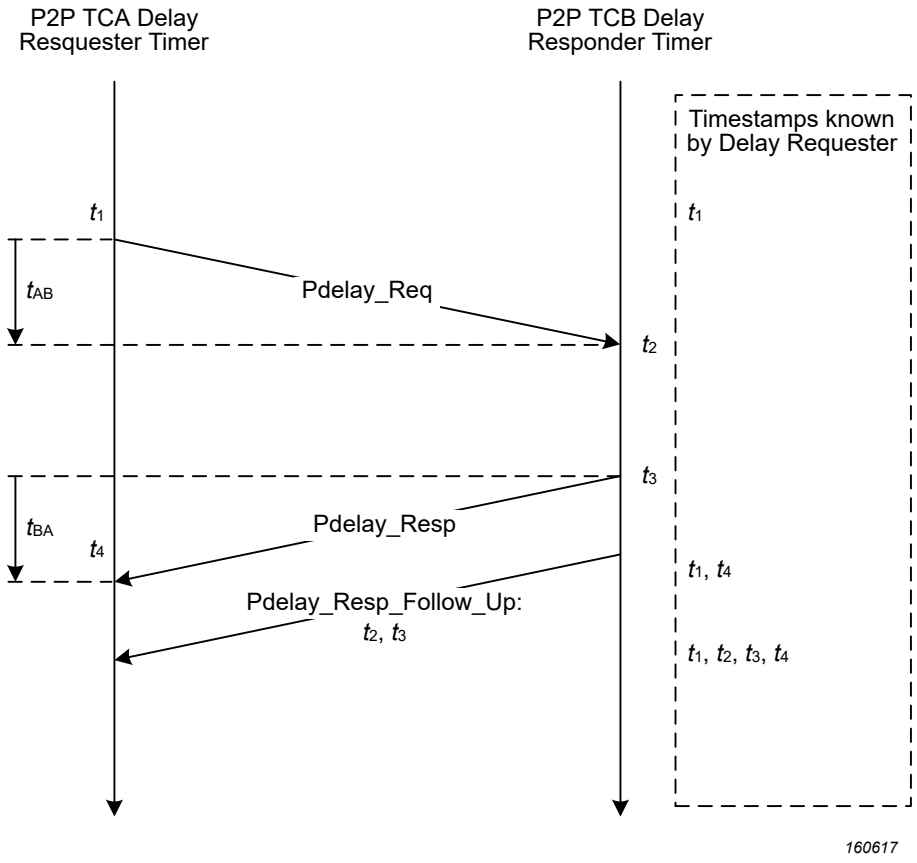


Figure 416. Propagation delay calculation in clocks supporting peer-to-peer path correction

The propagation delay is calculated in the following way:

1. Port-1 issues a Pdelay_Req message and generates a timestamp, t1, for the Pdelay_Req message.
2. Port-2 receives the Pdelay_Req message and generates a timestamp, t2, for this message.
3. Port-2 returns a Pdelay_Resp message and generates a timestamp, t3, for this message.

To minimize errors because of any frequency offset between the two ports, Port-2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. The Port-2 returns any one of the following:

- The difference between the timestamps t2 and t3 in the Pdelay_Resp message.
- The difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message.
- The timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.

4. Port-1 generates a timestamp, t4, on receiving the Pdelay_Resp message.
5. Port-1 uses all four timestamps to compute the mean link delay.

70.3.8.4 Frequency range of the reference timing clock

The timestamp information is transferred across asynchronous clock domains, that is, from MAC clock domain to application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is 4 clock cycles of II and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second frame.

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (1 ns resulting in 1 GHz) and the timing constraints achievable for logic operating on the PTP clock. In addition, the resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes. Because the MII clock frequency is fixed by IEEE specification, the minimum PTP clock frequency required for proper operation depends upon the operating mode and operating speed of the MAC as shown in Table 4-1.

Table 647. Minimum PTP clock frequency cycle

Mode	Minimum gap between two SFDs	Minimum PTP frequency
100-Mbps full-duplex operation	+ 24 clocks of min IFG + 16 clocks of preamble) 168 MII clocks (128 clocks for a 64-byte frame	(3 * PTP) + (4 * MII) ≤ 168 * MII that is, ~0.5 MHz ((168 – 4) * 40 ns ÷ 3 = 2,180 ns period)

70.3.8.5 PTP processing and control

shows the common message header for the PTP messages. This format is taken from IEEE standard 1588-2008 (Revision of IEEE Std. 1588-2002).

Table 648. Message format defined in IEEE 1588-2008

Bits									
7	6	5	4	3	2	1	0	OCTETS	OFFSET
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6

Table continues on the next page...

Table 648. Message format defined in IEEE 1588-2008 (continued)

Bits									
7	6	5	4	3	2	1	0	OCTETS	OFFSET
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField ()								1	32
logMessageInterva								1	33

There are some fields in the Ethernet payload that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP frames:

- PTP Frames Over IPv4
- PTP Frames Over IPv6
- PTP Frames Over Ethernet

70.3.8.5.1 PTP frames over IPv4

The table below provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in End-to-end transparent clock.

Table 649. IPv4-UDP PTP frame fields required for control and status

Field matched	Octet position	Matched value	Description
MAC Frame Type	12, 13	0x0800	IPv4 datagram
IP version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed. 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP-Primary multicast address: 224.0.1.129 PTP-Pdelay multicast address: 224.0.0.107
UDP Destination Port	36, 37	0x013F, 0x0140	0x013F – PTP event message [1] 0x0140 – PTP general messages
PTP control Field (IEEE version 1)	74	0x00/0x01/0x02/0x03/0x04	0x00 – SYNC, 0x01 – Delay_Req

Table continues on the next page...

Table 649. IPv4-UDP PTP frame fields required for control and status (continued)

Field matched	Octet position	Matched value	Description
			0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management
PTP Message Type Field (IEEE version 2)	42 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xB/0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only).

70.3.8.5.2 PTP Frames Over IPv6

The table below provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in End-to-end transparent clock.

Table 650. IPv6-UDP PTP frame fields required for control and status

Field Matched	Octet Position	Matched Value	Description
MAC Frame Type	12, 13	0x86DD	IP datagram
IP version	14(bits 7:4)	0x6	IP version is IPv6
Layer 4 Protocol	20 [1]	0x11	UDP
PTP Multicast Address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP – Primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP – Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57 [1]	0x013F, 0x140	0x013F – PTP event message 0x0140 – PTP general messages

Table continues on the next page...

Table 650. IPv6-UDP PTP frame fields required for control and status (continued)

Field Matched	Octet Position	Matched Value	Description
PTP Control Field (IEEE version 1)	93 [1]	0x00/0x01/0x02/ 0x03/0x04	0x00 – SYNC 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management (version1)
PTP Message Type Field (IEEE version 2)	74 [1] (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/ 0xB/0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	75 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

1. The Extension Header is not defined for PTP packets.

70.3.8.5.3 PTP frames over Ethernet

The table below provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of the IEEE 1588-2008 standard and the message format defined in End-to-end transparent clock.

Table 651. Ethernet PTP frame fields required for control and status

Field Matched	Octet Position	Matched Value	Description
MAC Destination Multicast Address[1]	0-5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses[2]: 01-1B-19-00-00-00 01-80-C2-00-00-0E[3]
MAC Frame Type	12, 13	0x88F7	PTP Ethernet frame
PTP Control Field (IEEE version 1)	45	0x00/0x01/0x02/ 0x03/0x04	0x00 – SYNC 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management

Table continues on the next page...

Table 651. Ethernet PTP frame fields required for control and status (continued)

Field Matched	Octet Position	Matched Value	Description
PTP Message Type Field (IEEE version 2)	14 (nibble)	0x0/0x1/0x2/0x3/0x8/ 0x9/0xB/0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	14 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

1. The address match of destination addresses (DA) programmed in MAC address 1 to 31 is used if the control bit 18 (TSENMACADDR: enable MAC address for PTP frame filtering) of the timestamp control register is set.
2. IEEE standard 1588-2008, Annex F
3. The Ethernet controller does not consider the PTP version 1 messages with peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

70.3.8.6 Transmit path functions

The MAC captures a timestamp when the start frame delimiter (SFD) of a frame is sent on MII. The frames for which you want to capture timestamps are controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether a timestamp should be captured for that frame.

The MAC does not process the transmitted frames to identify the PTP frames. You need to specify the frames for which you want to capture timestamps.

Use the control bits in the transmit descriptor (see). The MAC returns the timestamp to the software inside the corresponding transmit descriptor, thus connecting the timestamp automatically to the specific PTP frame. The 64-bit timestamp information is written to the TDES6 and TDES7 fields. The TDES7 field holds the 32 least significant bits of the timestamp.

70.3.8.7 Receive path functions

The MAC captures the timestamp of all frames received on the MII. The MAC does not process the received frames to identify the PTP frames in the default mode, that is, when the advanced timestamp feature is not selected.

The MAC can be programmed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. You can control the snapshot of the time to be sent to the application by using the following options of the MAC_Timestamp_Control register.

- Enable snapshot for all packets.
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp.
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet.
- Enable timestamp snapshot for the received packet for IPv4 or IPv6.
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP).

- Enable the node to be a master or slave and select the snapshot type This feature controls the type of messages for which snapshots are taken.

The DMA returns the timestamp to the software inside the corresponding receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in normal descriptor RDES1 and the snapshot of the timestamp is written in RDES0 and RDES1 fields of context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

70.3.8.8 IEEE 1588-2008 advanced timestamps

In addition to the basic timestamp features (see) the Ethernet controller supports the following advanced timestamp features defined in the IEEE 1588-2008 standard:

- Supports the IEEE 1588-2008 (version 2) timestamp format.
- Provides an option to take snapshots of all frames or only PTP type frames.
- Provides an option to take snapshots of only event messages.
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end, and peer-to-peer.
- Provides an option to select the node to be a master or slave for ordinary and boundary clock.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to measure sub-second time in digital or binary format.

70.3.8.8.1 Reference timing source

The Ethernet controller supports the 48-bit seconds field reference timing sources featured in the IEEE 1588-2008 standard.

The Ethernet controller supports 80-bit timestamping. The timestamp has the following fields:

- UInteger48 secondsField

  The seconds field is the integer portion of the timestamp in units of seconds and is 48-bits wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000 0000 0002.

- UInteger32 nanosecondsField

  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000 0001.

  The nanoseconds field supports the following two modes:

  — Digital rollover mode: The maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.

  — Binary rollover mode: The nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit.

You can set these modes by using the bit 9 (TSCTRLSSR) of MAC_TIMESTAMP_CONTROL.

When you select the advanced timestamp feature, the timestamp maintained in the core is still 64-bit wide. The overflow to the upper 16-bits of seconds register happens once in 130 years. You can read the values of the upper 16-bits of the seconds field from the CSR register.

70.3.8.9 System time register module

The 64-bit time is maintained in this module and updated using the input reference clock.

This time is the source for taking snapshots (timestamps) of Ethernet frames being transmitted or received at the MII.

The system time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is written with the value in the timestamp update registers, while for system time correction, the offset value is added to or subtracted from the system time.In the fine correction method, a slave clock's frequency drift with respect to the master clock (as defined in IEEE 1588) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP sync message intervals. In this method, an accumulator sums up

the contents of the addend register, as shown in Figure 4-2. The arithmetic carries that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider.

This algorithm is shown in :

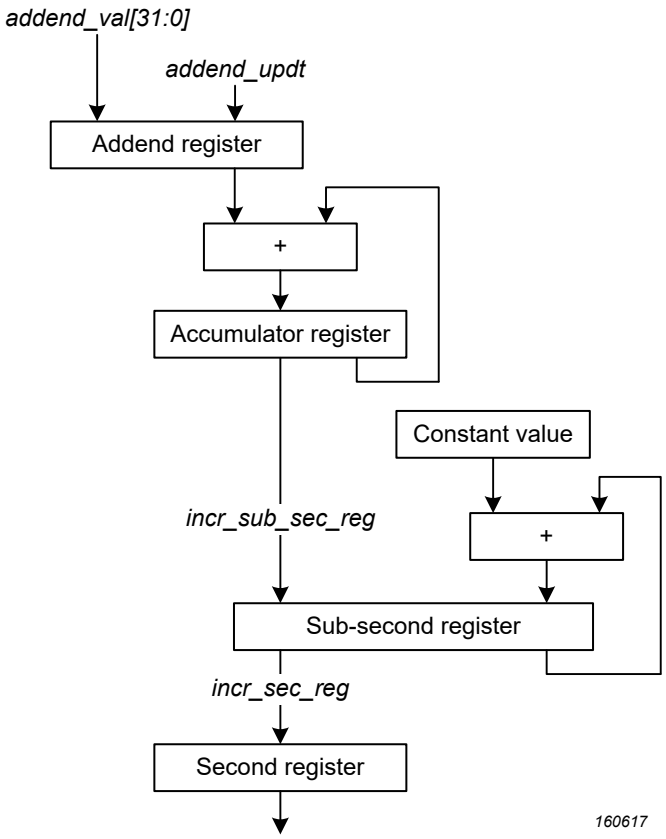


Figure 417. System update using fine method

The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. Hence, if the reference clock is, for example, 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Hence, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, to 65 MHz for example, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC4EC4. If the clock drifts higher, to 67 MHz for example, the addend register must be set to 0xBF0B 7672. When the clock drift is nil, the default addend value of 0xC1F0 7C1F (2^{32} / 1.32) must be programmed.

In , the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the sync messages and update the addend register accordingly.

Initially, the slave clock is set with FreqCompensationValue0 in the addend register. This value is:

FreqCompensationValue0 = 2^{32} / FreqDivisionRatio

The frequency division (FreqDivisionRatio) is the ratio of the reference clock frequency to the required clock frequency.

If MasterToSlaveDelay is initially assumed to be the same for consecutive sync messages, the algorithm described below must be applied. After a few sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- At time $MasterSyncTime_n$ the master sends the slave clock a sync message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as:

  $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$

- The master clock count for current sync cycle, $MasterClockCount_n$ is given by:

  $MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$ (assuming that MasterToSlaveDelay is the same for sync cycles n and n - 1)

- The slave clock count for current sync cycle, $SlaveClockCount_n$ is given by:

  $SlaveClockCount_n = SlaveClockTimen_n - SlaveClockTime_{n-1}$

- The difference between master and slave clock counts for current sync cycle, $ClockDiffCount_n$ is given by:

  $ClockDiffCount_n = MasterClockCount_n - SlaveClockCount_n$

- The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is given by:

  $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$

- The frequency compensation value for addend register, $FreqCompensationValue_n$ is given by:

  $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one sync cycle; however, it may take several cycles, because of changing network propagation delays and operating conditions.

This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm corrects it at the cost of more sync cycles.

70.3.9 DMA controller description

The DMA has independent transmit and receive engines and a CSR space. The transmit engine transfers data from the system memory to the device port (MTL), while the receive engine transfers data from the device port to the system memory. The DMA controller uses descriptors to efficiently move data from source to destination with minimal host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the host CPU for situations such as frame transmit and receive transfer completion, and other normal/error conditions.

The DMA and the host driver communicate through two data structures:

- Control and status registers (CSR).
- Lists and data buffers.

The DMA transfers data frames received by the MAC to the receive buffer in the system memory, and transmit data frames from the transmit buffer in the system memory. Descriptors that reside in the system memory act as pointers to these buffers. There are two descriptor lists; one for reception, and one for transmission.The DMA supports up to two Tx and two Rx descriptor lists (or DMA channels). The base address of each list is written to the respective Tx descriptor list address register. The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. This offset can be controlled by the DSL field of DMA_CH0_CTRL/DMA_CH1_CTRL registers.

The number of descriptors in the list is programmed in the respective Tx (or Rx) descriptor ring length register. Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the list address register to create a descriptor ring. The descriptors lists reside in the host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptors. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when the end-of-frame is detected. data chaining can be enabled or disabled.

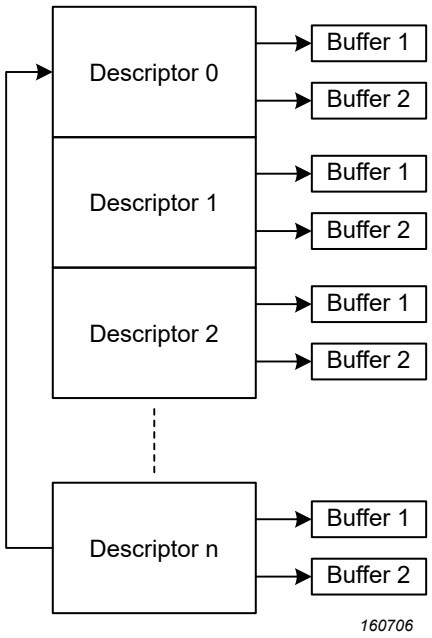


Figure 418. Ring structure

The transmit and receive engines enter the running state and attempt to acquire descriptors from the respective descriptor lists. The receive and transmit engines then begin processing receive and transmit operations. The transmit and receive processes are independent of each other and can be started or stopped separately.

70.3.9.1 Host bus burst access

The DMA attempts to execute fixed-length Burst transfers on the AHB master interface if configured to do so (FB bit of DMA system bus mode register 0). The maximum burst length is indicated and limited by the PBL field (DMA register 0[13:8]). The receive and transmit descriptors are always accessed in the maximum possible (limited by PBL or 16 x 8/bus width) burst size for the 16 bytes to be read.

The transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise (no fixed length burst), it transfers data using INCR (undefined length) and SINGLE transactions.

The receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in MTL Receive FIFO or when the end of frame (when it is less than the configured burst length) is detected in the Receive FIFO. The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. If the end of frame is reached before the fixed-burst ends on the AHB interface, then dummy transfers are performed to complete the fixed burst. Otherwise (FB bit of DMA_ SYSBUS_MODE register is reset), it transfers data using INCR (undefined length) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the AHB initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16) because the AHB interface does not support more than INCR16.

70.3.9.2 Host data buffer alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start addresses for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers

with addresses aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example: buffer read

If the transmit buffer address is 0x0000 0FF2 (for 32-bit data bus), and 15 bytes need to be transferred, then the DMA reads five full words from address 0x0000 0FF0, but when transferring data to the MTL Tx Queue, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures it transfers a full 32-bit data to the MTL Transmit FIFO unless it is the end-of-frame.

Example: buffer write

If the receive buffer address is 0x0000 0FF2 (for 64-bit data bus) and 16 bytes of a received frame need to be transferred, then the DMA writes 3 full words from address 0x0000 0FF0. But the first 2 bytes of the first transfer and the last 6 bytes of the third transfer have dummy data.

70.3.9.3 Buffer size calculations

The DMA does not update the size fields in the transmit and receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES2) towards the MAC core. If a descriptor is marked as first (FS bit of TDES2 is set), then the DMA marks the first transfer from the buffer as the start of the frame. If a descriptor is marked as last (LS bit of TDES2), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. When the FS bit of a descriptor is set, the amount of valid data in a buffer is accurately indicated by the buffer size field (programmed in DMA channel receive control register) minus the data buffer pointer offset. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, the buffer may not be full (as indicated by the buffer size in bits 14:1 of receive control register). To compute the amount of valid data in this final buffer, the driver must read the packet length (FL bits of RDES3[14:0]) and subtract the sum of the buffer sizes of the preceding buffers in this packet. The Rx DMA always transfers the start of the next packet with a new descriptor.

Even when the start address of a receive buffer is not aligned to the system bus's data width, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the receive descriptor to have a 0x1002 offset. The receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes because of the start address offset.

70.3.9.4 DMA arbiter

The arbiter inside the DMA module performs the arbitration between the transmit and receive channel accesses to the AHB host interface. Two types of arbitrations are possible: round-robin, and fixed-priority.

When round-robin arbitration is selected (DA bit of DMA mode register is reset), the arbiter allocates the data bus in the ratio set by the PR bits of DMA mode register, when both transmit and receive DMAs are requesting for access simultaneously. When the DA bit is set, the receive DMA always gets priority over the transmit DMA for data access by default. When the TXPR bit (bit 11 of DMA mode register) is also set, then the transmit DMA gets priority over the receive DMA.

70.3.9.5 Transmission

70.3.9.5.1 TxDMA operation: Default (non-OSF) mode

The transmit DMA engine in default mode proceeds as follows:

1. The host sets up the transmit descriptor (TDES0-TDES3) and sets the OWN bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet frame data.
2. The application advances the descriptor tail pointer offset value of the transmit channel.
3. While in the run state, the DMA runs an arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended and bit 2 and bit 16 of the status register of the corresponding DMA channel are set and the Tx engine proceeds to step 11.
   - The descriptor is flagged as owned by the application (TDES3[31] = 0).
   - The descriptor tail pointer is equal to the current descriptor pointer in ring descriptor list mode.
   - An error condition occurs.
6. If the acquired descriptor is flagged as owned by DMA (TDES3[31] = 1), the DMA decodes the transmit data buffer address from the acquired descriptor.
7. The DMA fetches the transmit data from the host memory and transfers the data to the MTL for transmission.
8. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3, 4, and 5 are repeated until the end-of-Ethernet-frame data is transferred to the MTL.
9. When frame transmission is complete, if IEEE 1588 timestamping was enabled for the frame (as indicated in the transmit status) the timestamp value obtained from MTL is written to the transmit descriptor (TDES0 and TDES1) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES3). Because the OWN bit is cleared during this step, the host now owns this descriptor. If timestamping was not enabled for this frame, the DMA does not alter the contents of TDES0 and TDES1.
10. Bit 0 of the status register of the corresponding channel is set after completing transmission of a packet that has interrupt on completion (TDES2[31]) set in its last descriptor. The DMA engine returns to Step 3.
11. when it receives a transmit poll demand and the underflow interrupt status bit is cleared. If the application stopped the DMA by clearing bit 0 of the transmit control register of the corresponding DMA channel, the DMA enters the stop state.

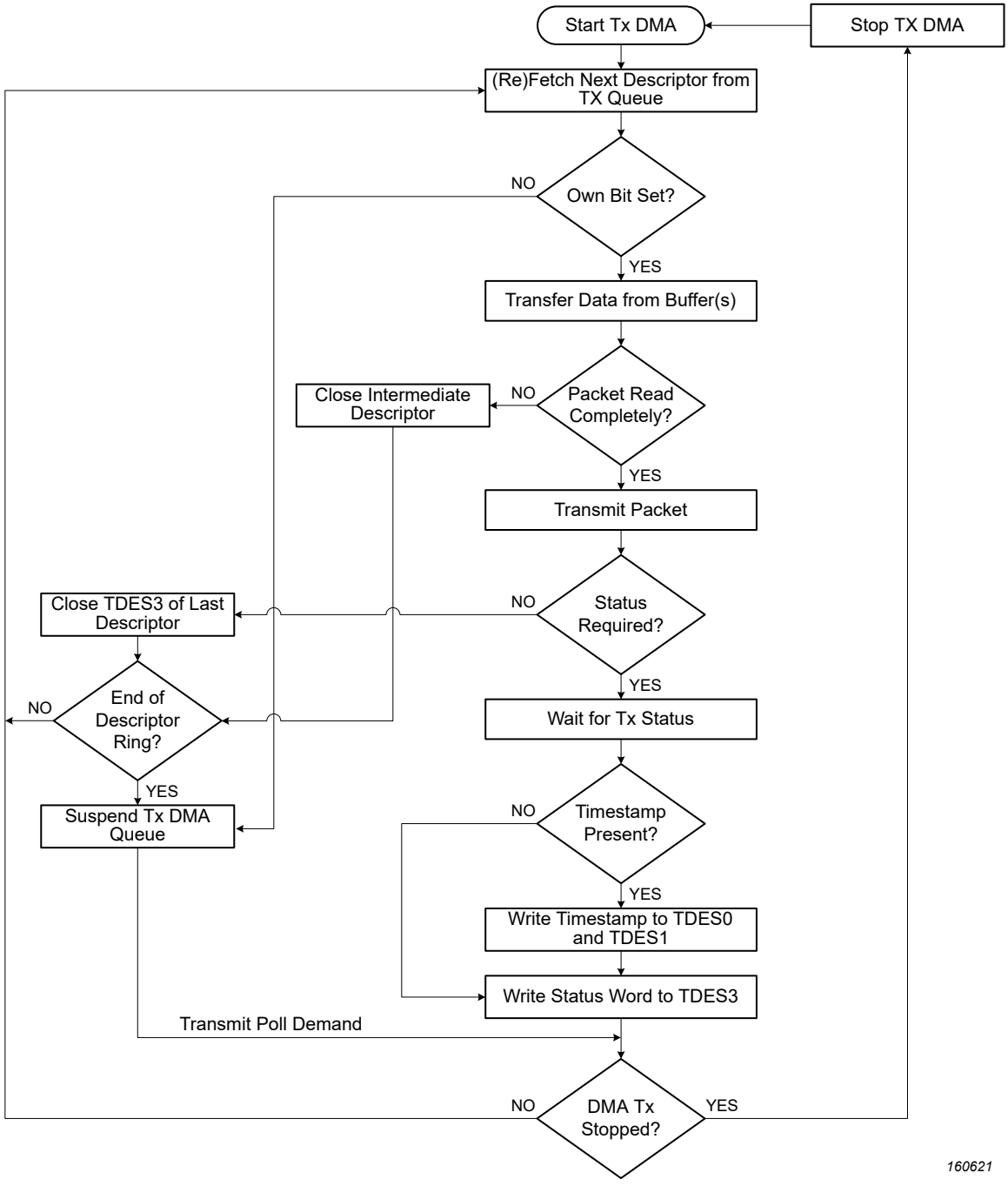


Figure 419. Transmit DMA operation in default mode

70.3.9.5.2 TxDMA operation: OSF mode

In the run state, if bit 4 is set in the transmit control register of the corresponding DMA channel, the transmit process can simultaneously acquire two packets without closing the status descriptor of the first frame. As the transmit process finishes transferring the first frame, it immediately polls the transmit descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

In OSF mode, the run state transmit DMA operates in the following sequence:

1. The DMA operates as described in steps 1 to 7of the TxDMA (default mode).

2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into suspend mode and skips to Step 7.
4. The DMA fetches the transmit frame from the host memory and transfers the frame to the MTL until the end-of-frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's frame transmission status and timestamp. Once the status is available, the DMA writes the timestamp to TDES0 and TDES1, if such timestamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared OWN bit, to the corresponding TDES3, thus closing the descriptor. If timestamping was not enabled for the previous frame, the DMA does not alter the contents of TDES2 and TDES3.
6. If enabled, the transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to Step 3 (when status is normal). If the previous transmission status shows an underflow error, the DMA goes into suspend mode (Step 7).
7. In suspend mode, if a pending status and timestamp are received from the MTL, the DMA writes the timestamp (if enabled for the current frame) to TDES2 and TDES3, then writes the status to the corresponding TDES3. It then sets relevant interrupts and returns to suspend mode. If no status is pending and the application stopped the DMA by clearing bit 0 of the transmit control register of the corresponding DMA channel, the DMA enters the stop state.
8. The DMA can exit suspend mode and enter the run state (go to Step 1 or Step 2 depending on pending status) only after receiving a transmit poll demand in transmit descriptor tail pointer register of the corresponding channel.

As the DMA fetches the next descriptor in advance before closing the current descriptor, the descriptor chain should have more than 2 different descriptors for correct and proper operation.

The basic flow is described in the figure below.

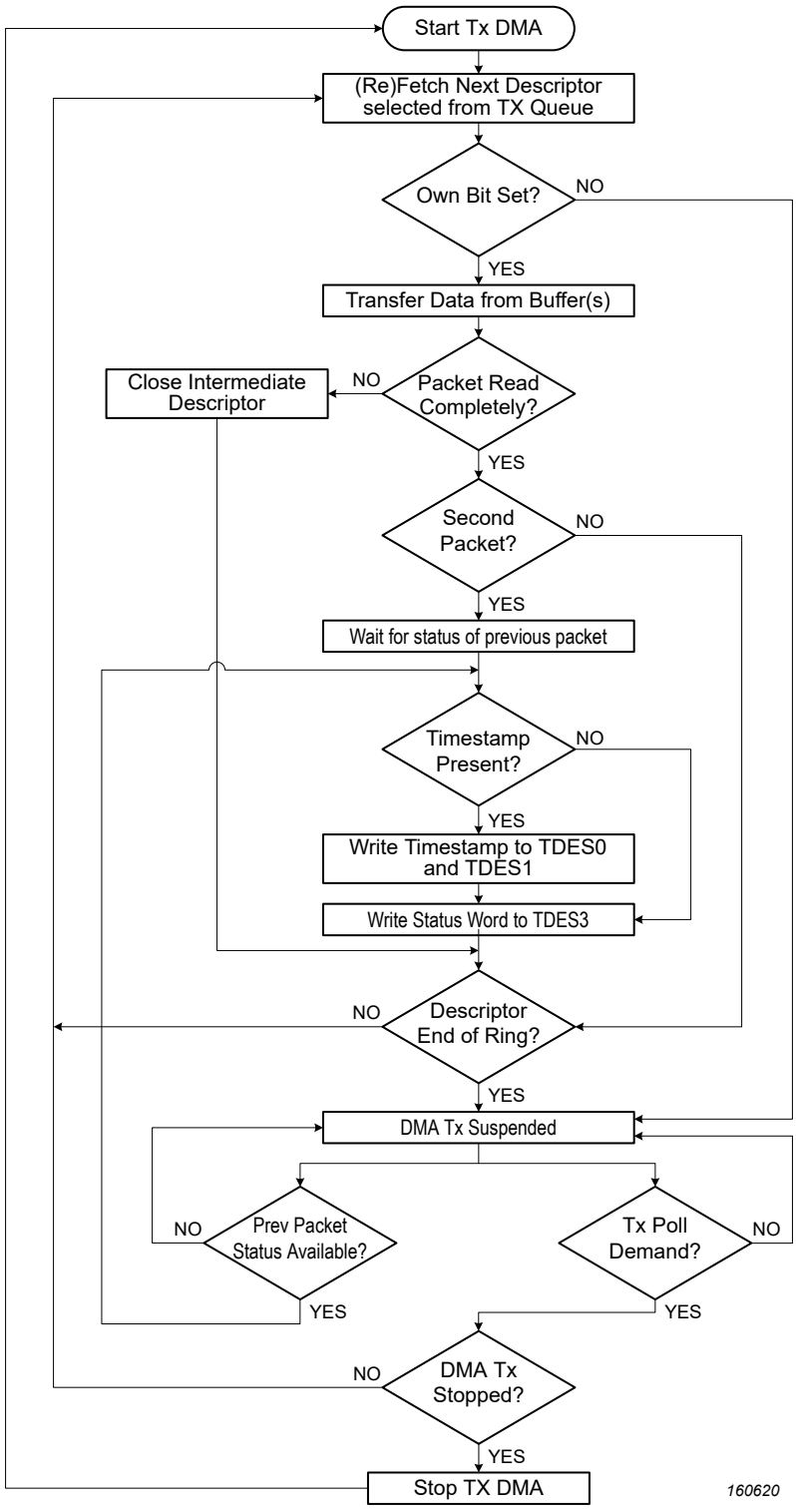


Figure 420. Transmit DMA operation in OSF mode

70.3.9.5.3 Timestamp correction

According to the IEEE 1588 specification, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the

network. Because the reference timing source (the PTP clock clk_ptp_ref_i) is different from the MAC Tx or Rx clock, the captured timestamp must be corrected for latency issues because of synchronization. In addition, latency issues between the internal snapshot point and the recommended capture point (the boundary between the node and the network), must also be corrected.

Ingress Correction: On the receive side, the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (ingress correction value) programmed in the ingress correction register. The value that needs to be programmed in the ingress correction register is calculated as follows:

The timestamp correction because of synchronization is compensated by adding INGRESS_SYNC_CORR to the synchronized timestamp value as:

INGRESS_SYNC_CORR = - (2*PTP_CLK_PER)

The latency correction between the message timestamp point and the internal timestamp snapshot point is done by subtracting the latency value(INGRESS_LATENCY) with the captured timestamp as:

Ingress Correction = INGRESS_SYNC_CORR - INGRESS_LATENCY

Ingress correction is performed by programming the TSIC field in the MAC timestamp ingress correction register. The ingress correction is always negative and has a unit of nanoseconds. The value is represented in complement form as follows:

When the TSCTRLSSR bit in the MAC_Timestamp_Control register is set, this has an accuracy of 1 ns. It is represented by setting bit 31 to 1 and bits 30:0 containing

10^9 - <ingress_correction_value> represented in binary. For example, if the required correction value is -5 ns, then the programmed value is 0xBB9A_C9FB. When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. It is represented by setting bit 31 to 1 and bits 30:0 containing

2^{31} - <ingress_correction_value> represented in binary.

Egress correction:

On the transmit side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (egress correction value) programmed in the egress correction register. The value that needs to be programmed in the egress correction register is calculated as follows:

The timestamp correction because of synchronization is compensated by adding EGRESS_SYNC_CORR to the synchronized timestamp value as follows:

When enable one step timestamp feature is selected,

EGRESS_SYNC_CORR = (1*PTP_CLK_PER + 4*TX_CLK_PER)

Otherwise,

EGRESS_SYNC_CORR = -(2*PTP_CLK_PER)

The egress latency correction between the recommended capture point and the internal timestamp snapshot point is done by adding the latency value (EGRESS_LATENCY) with the captured timestamp as follows:

Egress Correction = EGRESS_SYNC_CORR + EGRESS_LATENCY

Egress correction is performed by programming the TSEC field in the MAC timestamp egress correction register. The egress correction can be positive or negative and has a unit of nanoseconds. Negative values are represented in complement form as follows:

When the TSCTRLSSR bit in the MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The value must not exceed 0x 3B9A_C9FF. If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 10^9 - <egress_correction_value> represented in binary.

For example, if the required correction value is -5 ns, then the programmed value should be 0xBB9A_C9FB

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 nS. If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The

maximum value is 0x7FFF_FFFF. If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 2^{31} - <egress_correction_value> represented in binary.

70.3.9.5.4 Transmit frame processing

The transmit DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The DA, SA, and type/len fields contain valid data. If the transmit descriptor indicates that the MAC core must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes.

Frames can be data-chained and can span several buffers. Frames must be delimited by the first descriptor (TDES3[29]) and the last descriptor (TDES3[28]), respectively.

As transmission starts, the first descriptor must have (TDES3[29]) set. When this occurs, frame data transfers from the host buffer to the MTL Tx Queue. Concurrently, if the current frame has the last descriptor (TDES3[28]) clear, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES3[29] clear. If TDES3[28] is clear, it indicates an intermediary buffer. If TDES3[28] is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the transmit descriptor 3(TDES3) word of the descriptor that has the last descriptor bit set in transmit descriptor 3 (TDES3[28]). At this time, if the interrupt on completion (TDES2[31]) was set, bit 0 of the status register of the corresponding DMA channel is set, the next descriptor is fetched, and the process repeats.

The actual frame transmission begins after the MTL Tx Queue has reached either a programmable transmit threshold (DMA operation mode register, bits 6:4), or a full frame is contained in the FIFO. There is also an option for store and forward mode (MTL operation mode register, bit 1). In this mode, the descriptors are released (OWN bit TDES0[31] clears) when the DMA finishes transferring the frame.

To ensure proper transmission of a frame and the next frame, you must specify a non-zero buffer size for the transmit descriptor that has the last descriptor (TDES3[28]) set.

70.3.9.5.5 Transmit polling suspended

Transmit polling can be suspended by either of the following conditions:

- The DMA detects a descriptor owned by the host (TDES3[31]=0). To resume, the driver must give descriptor ownership to the DMA and then issue a poll demand command by writing the tail pointer register.
- A frame transmission is aborted when a transmit error because of underflow is detected. The appropriate transmit descriptor 3 (TDES3) bit is set.
- The DMA detects that the tail pointer is equal to the current descriptor closed by it. To resume, the software driver must modify the tail pointer register.

If the second condition occurs, bit 14 of the status register of the corresponding DMA channel is set.

In all three cases, the position in the transmit list is retained. The retained position is that of the descriptor following the last descriptor closed by the DMA.

The driver must explicitly issue a transmit poll demand command after rectifying the suspension cause.

70.3.9.5.6 Transmit channel arbitration

An arbiter provides access to multiple DMAs trying to access the bus interface unit (BIU). Figure 3-21 shows the arbitration process.

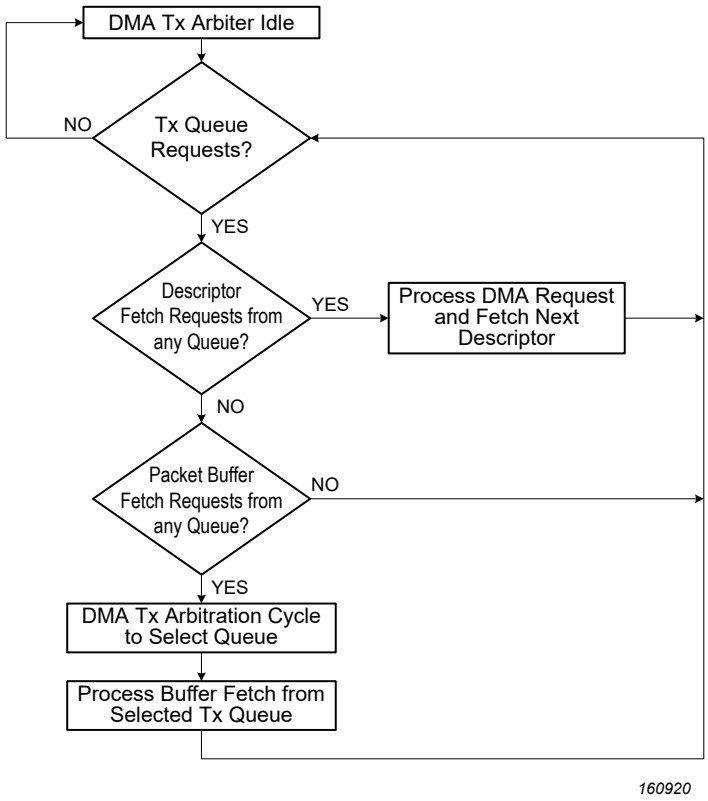


Figure 421. DMA transmit channel arbitration process

When there is any request in the Tx Queue, the DMA arbiter checks the type of the request: packet buffer fetch or descriptor fetch request. The descriptor fetch requests have higher priority than the buffer requests. Therefore, when there is a descriptor fetch request, the DMA arbiter acknowledges the DMA channel that is requesting a descriptor fetch. If there is no descriptor fetch request, the arbiter looks for packet buffer fetch requests.

The DMA arbiter acknowledges the descriptor fetch request of one DMA channel at a time. Descriptor fetch requests are granted using a fixed priority with the higher channel having higher priority (channel 1 having priority over channel 0, channel 2 having priority over channel 1 and so on). For packet buffer fetches, the DMA arbiter uses the programmed channel weight and priority to decide which channel to acknowledge. The DMA arbiter performs a burst-by-burst arbitration based on one of the following algorithms:

- Weighted strict priority (WSP): In WSP arbitration mode, the arbiter first processes channel 7 (or the last enabled channel) and then channel 6, channel 5, and so on. If any channel does not have a frame to transmit, the weight of that channel gets reassigned to channel7 (or last enabled channel). If channel 7 has no frames to transmit, the remaining weight is assigned to channel 6 and so on.
- Weighted round-robin (WRR): In WRR arbitration mode, the arbiter first selects the channel with the highest weight programmed, and then the channel with the next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel gets equally distributed to all channels that have frames to transmit.
- Fixed priority (FP): In fixed priority mode, channel 0 has the lowest priority and the last selected channel has the highest priority. The weight programmed in the transmit control register of a channel is ignored.

In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

70.3.9.5.7 Reception

In the receive path, the DMA reads a packet from the MTL Rx Queue and writes it to the frame data buffers of the corresponding DMA channel. The receive DMA engine's reception sequence is shown in and proceeds as follows:

1. The host sets up receive descriptors (RDES0-RDES3) and sets the OWN bit (RDES3[31]). The host must set the correct value in the receive descriptor tail pointer register of the corresponding DMA channel.
2. is set, the DMA enters the run state. While in the run state, the DMA polls the receive descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the host), the DMA enters the suspend state and jumps to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of the current descriptor.
5. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
6. Incoming frames are processed and placed in the acquired descriptor's data buffers.
7. The DMA takes the status of the received frame from the MTL and writes the status word to the current descriptor with the OWN bit cleared and the last descriptor bit set.
8. The DMA writes the frame length to RDES3. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. If IEEE 1588 timestamp feature is enabled, the DMA stores the timestamp (if available).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3; otherwise, go to the suspend state (step 11).
11. The receive DMA exits the Suspend state when a receive poll demand is given and the application advances the receive tail pointer register of a channel. The engine proceeds to step 2 and fetches the next descriptor.

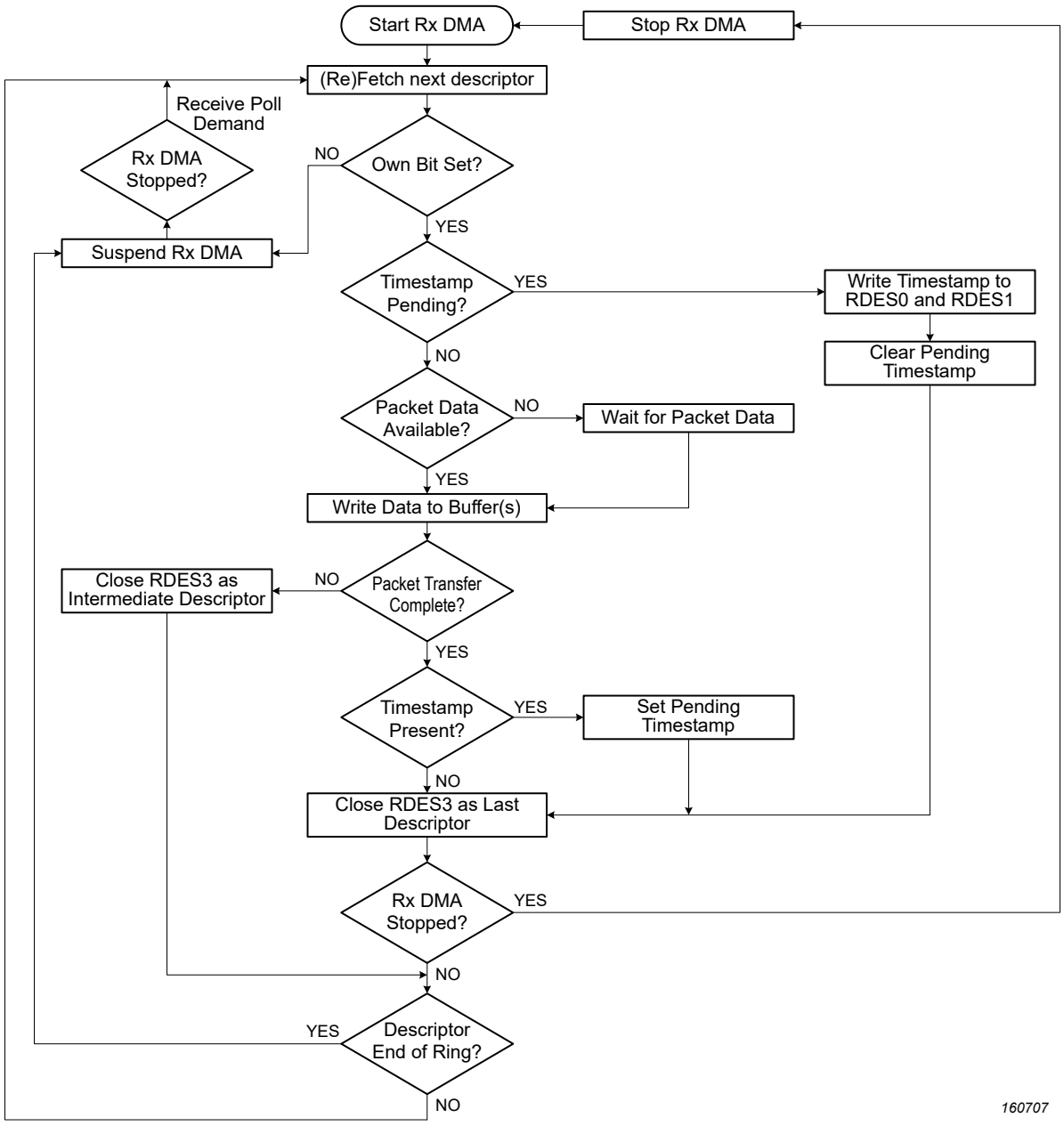


Figure 422. Receive DMA operation

70.3.9.5.8 Receive descriptor acquisition

The receive engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- It is set immediately after being placed in the run state.
- The descriptor tail pointer register value is ahead of the current descriptor acquired by the Rx DMA.
- The controller has completed frame reception, but the current receive descriptor is not yet closed.
- A receive poll demand is issued.

70.3.9.5.9 Receive frame processing

The MAC transfers the received frames to the host memory only when the frame passes the address filter and frame size is greater than or equal to configurable threshold bytes set for the Receive FIFO of MTL, or when the complete frame is written to the queue in store and forward mode.

If the frame fails the address filtering, it is dropped in the MAC block itself (unless receive all bit 31 is set in the MAC frame filter register;). Frames that are shorter than 64 bytes, because of the collision or premature termination, can be purged from the MTL Rx Queue.

When the DMA application interface AHB becomes ready, it transfers the data and sets the following: If the packet fits in a single descriptor, the DMA sets both the last descriptor (RDES3[28]) and the first descriptor (RDES3[29]). If the packets fit into more than one descriptor, the DMA sets the first descriptor (RDES3[29]) to delimit the packet. The descriptors are released when the OWN (RDES[31]) bit is reset to 0, either as the data buffer fills up or as the last segment of the frame is transferred to the receive buffer. The received packets status is updated in the last descriptor.

If interrupt enabled on completion (RDES3[30]) bit is set in any of the descriptors between the first and last descriptor of the packet and bit 6 of interrupt enable register of corresponding DMA channel is set, the DMA sets bit 6 of status register of corresponding DMA channel. The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host or when there are no more descriptors in the ring. When the DMA finds a descriptor owned by the application and if bit 7 of interrupt enable register of the corresponding DMA channel is set, the receive process sets bit 7 of status register of the corresponding DMA channel and then enters the suspend state. The position in the receive list is retained.

70.3.9.5.10 Interrupts

Interrupts can be generated as a result of various events. The DMA interrupt status register contains an enable bit for each of the events that can cause an interrupt.

There are two groups of interrupts, normal and abnormal, as described in the DMA status channel status register The normal group is for events that happen during the normal transfer of packets (TI, RI, TBU) while the abnormal interrupt events are for error events. Interrupts are cleared by writing a 1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal is de-asserted. Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated.

For example, receive interrupt bit 6 of the DMA channel status register indicates that one or more packets were transferred to the application buffer. The driver must scan all descriptors, from the last recorded position to the first one, owned by the DMA to determine how many packets are received. The interrupt pin is asserted because of any event in this status register only if the corresponding interrupt enable bit is set in DMA interrupt enable register.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the DMA interrupt status register for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs after the driver has cleared the appropriate bit in the DMA status register.

70.3.9.5.11 Error response to DMA

For any data transfer initiated by a DMA channel, if the slave replies with an error response, that DMA stops all operations and updates the error bits and the fatal bus error bit in the status register of corresponding DMA channel. That DMA controller can resume operation only after soft resetting or hard resetting the core and re-initializing the DMA. The rest of the DMA channels are not affected by such errors.

70.3.10 Ethernet descriptors

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory. The descriptors supported by Ethernet module are described in subsequent sections.

Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit for number of descriptors that can be used for a single packet.

70.3.10.1 Descriptor structure

Ethernet module supports the ring structure for DMA descriptor.

In Ring structure, descriptors are separated by the word, dword, or lword number programmed in the DSL field of the DMA channel control register. The application needs to program the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

- DMA channel transmits descriptor ring length register(DMA_CH0_TXDESC_RING_LENGTH and DMA_CH1_TXDESC_RING_LENGTH).
- DMA channel receives descriptor ring length register (DMA_CH0_RX_CONTROL2 and DMA_CH1_RX_CONTROL2).

The descriptor tail pointer register contains the pointer to the descriptor address (N). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

Current descriptor pointer == descriptor tail pointer;

The DMA goes into the suspend mode when this condition occurs. The application must perform a write to the descriptor tail pointer register and update the tail pointer so that the following condition is true:

Current descriptor pointer < descriptor tail pointer;

The DMA automatically wraps around the base address when the end of ring is reached. See

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to descriptor base address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to descriptor base address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

The DMA writes the header length in RDES2 of the first receive descriptor (RDES3[29] (FD bit) is set) for the packet. The packet length is written in RDES3 of the last receive descriptor (RDES3[28] (LD bit) set). The buffer length for the payload is set by the driver through the RBSZ field in the corresponding DMA channel register 2 (receive control register). The DMA fills receive buffers fully in all except the last descriptor.

The header length is taken to be the value based on the bits programmed in the MAC extended configuration register HDSMS field (bits 22:20).

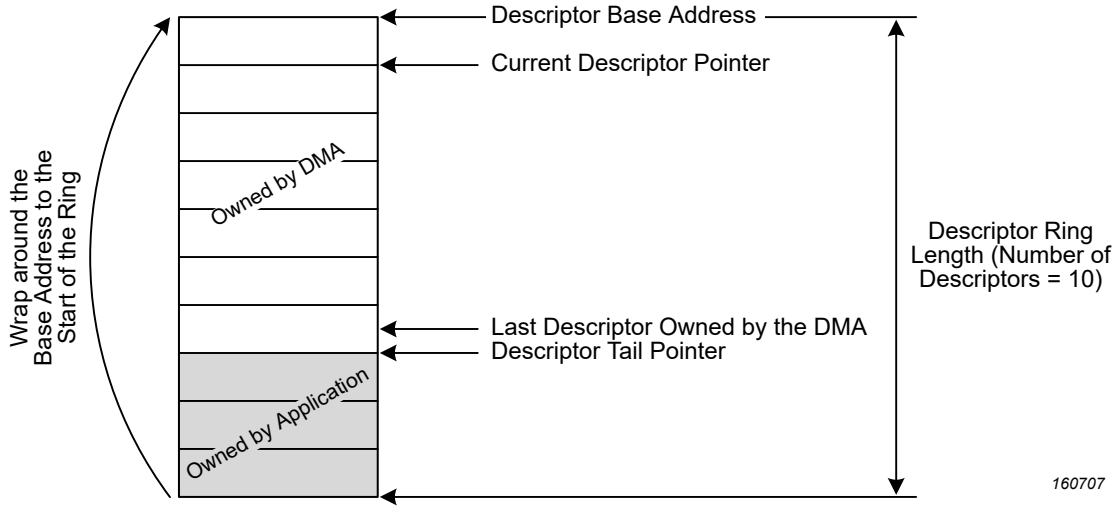


Figure 423. Descriptor ring

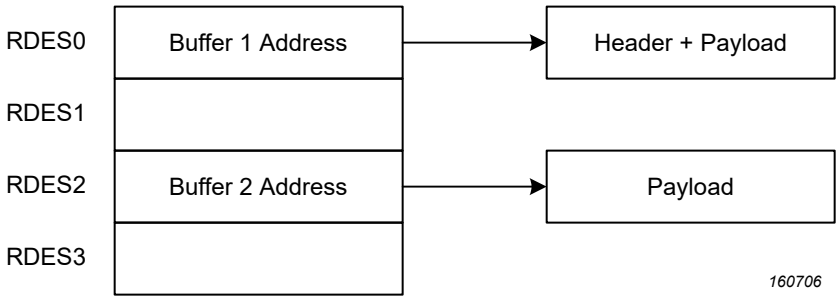


Figure 424. Descriptor structure

70.3.10.2 Descriptor endianness

The data bus can be configured for little-endian format.

70.3.10.3 Transmit descriptor

The DMA in Ethernet core requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields that can be used to control the MAC operation on per-transmit packet basis. The transmit normal descriptor has two formats: read format and write-back format.

70.3.10.3.1 Transmit normal descriptor (read format)

The data bus can be configured for either little-endian or big-endian format. The below figure shows the read format for a transmit normal descriptor.

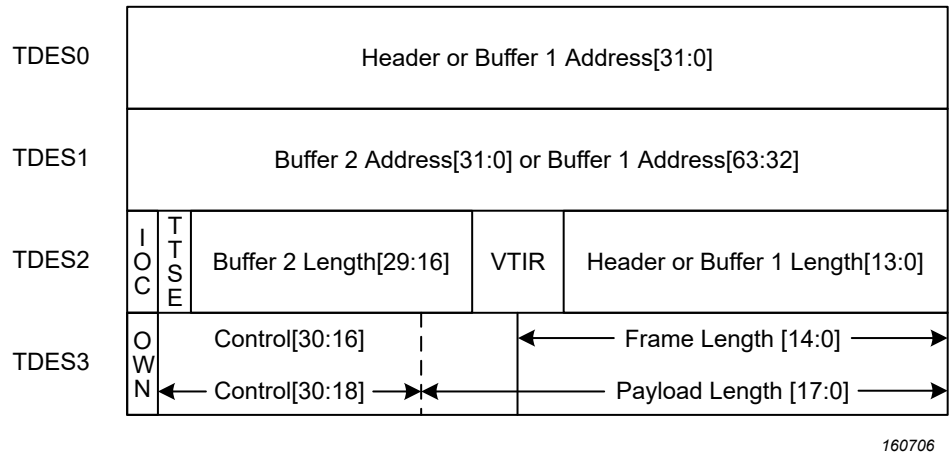


Figure 425. Transmitter descriptor read format

Table 652. Transmit descriptor word 0(TDES0)

Bit	Symbol	Description
31:0	BUF1AP	Buffer 1 address pointer or TSO header address pointer. These bits indicate the physical address of buffer 1. These bits indicate the TSO header address pointer when the following bits are set: TSE bit of TDES3 FD bit of TDES3

Table 653. Transmit descriptor word 1(TDES1)

Bit	Symbol	Description
31:0	BUF2AP	Buffer 2 or buffer 1 address pointer. This bit indicates the physical address of buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment. In 40- or 48-bit addressing mode, these bits indicate the most-significant 8- or 16- bits of the buffer 1 address pointer.

Table 654. Transmit descriptor word 2 (TDES2)

Bit	Symbol	Description
13:0	B1L	Buffer 1 length. This field is equal to buffer 1 length.
15:14	-	Reserved.
29:16	B2L	Buffer 2 length. The driver sets this field. When set, this field indicates buffer 2 length.

Table continues on the next page...

Table 654. Transmit descriptor word 2 (TDES2) (continued)

Bit	Symbol	Description
30	TTSE	Transmit timestamp enable. This bit enables the IEEE1588 timestamping for transmit packet referenced by the descriptor.
31	IOC	Interrupt on completion. This bit sets the TI bit in the DMA channel status register after the present packet has been transmitted.

Table 655. Transmit descriptor word 3 (TDES3)

Bit	Symbol	Description
14:0	FL	Frame length. This field is equal to the length of the frame to be transmitted in bytes. This field is equal to the total length of the frame to be transmitted: Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length
15	-	Reserved.
17:16	CIC	Checksum insertion control. These bits control the checksum calculation and insertion. The following list describes the bit encoding: • 0x0: Checksum insertion disabled. • 0x1: Only IP header checksum calculation and insertion are enabled. • 0x2: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. • 0x3: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware.
18	-	Reserved.
22:19	SLOTNUM	Slot number control bits in AV mode. These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field. DMA channel slot function control status. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels.
25:23	SAIC	SA Insertion Control These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must set the CRC Pad Control bits appropriately when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[24:23]:

Table continues on the next page...

Table 655. Transmit descriptor word 3 (TDES3) (continued)

Bit	Symbol	Description
		2'b00: Do not include the source address 2'b01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 2'b10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. 2'b11: Reserved These bits are valid in the EQOS-DMA, EQOS-AXI, and EQOS-AHB configurations when the Enable SA and VLAN Insertion on Tx option is selected while configuring the core and when the First Segment control bit (TDES3 [29]) is set. This field is valid only for the first descriptor.
27:26	CPC	CRC pad control. This field controls the CRC and pad insertion for Tx packet. This field is valid only when the first descriptor bit (TDES3[29]) is set. The following list describes the values of bits 27:26: • 0x0: CRC and pad insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. • 0x1: CRC insertion (disable pad insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the transmit buffer, that is, the packet being transferred from the transmit buffer is of length greater than or equal to 60 bytes. • 0x2: Disable CRC insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer. • 0x3: CRC replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer.
28	LD	Last descriptor. When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.
29	FD	First descriptor When this bit is set, it indicates that the buffer contains the first segment of a packet.
30	CTXT	Context type. This bit should be set to 0 for normal descriptor
31	OWN	Own bit. When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).

70.3.10.3.2 Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding transmit packet.

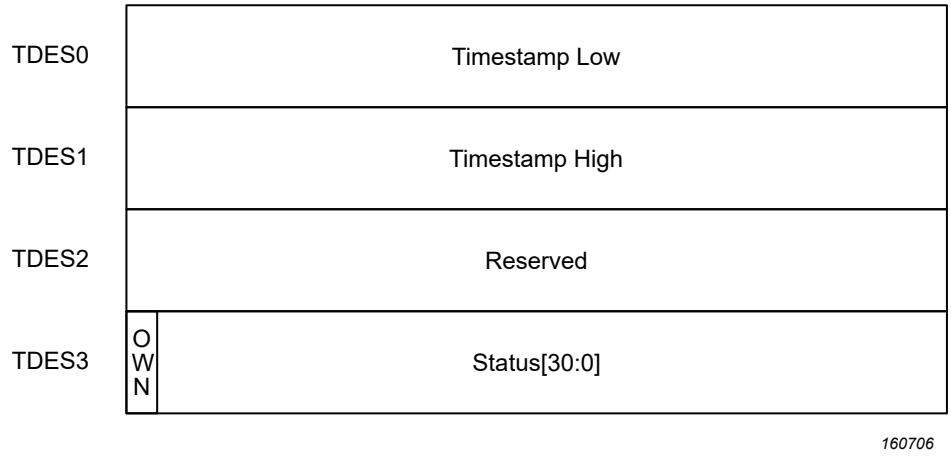


Figure 426. Transmitter descriptor write-back Format

70.3.10.3.3 TDES0 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 656. TDES0 normal descriptor (write-back Format)

Bit	Symbol	Description
31:0	TTSL	Transmit packet timestamp low. The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the last segment bit (LS) in the descriptor is set and the timestamp status (TTSS) bit is set.

70.3.10.3.4 TDES1 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 657. TDES1 normal descriptor (write-back format)

Bit	Symbol	Description
31:0	TTSH	Transmit packet timestamp high. The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the last segment bit (LS) in the descriptor is set and timestamp status (TTSS) bit is set.

70.3.10.3.5 TDES2 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 658. TDES2 normal descriptor (write-back format)

Bit	Symbol	Description
31:0	--	Reserved.

70.3.10.3.6 TDES3 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 659. TDES3 normal descriptor (write-back format)

Bit	Symbol	Description
0	IHE	IP header error. When IP header error is set, this bit indicates that the checksum offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet type field indicates an IPv4 payload.
1	DB	Deferred bit This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
2	UF	Underflow error. This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty transmit buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the MTL interrupt status register.
3	ED	Excessive deferral. This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times if DC bit is set in the MAC configuration register.
7:4	CC	Collision count/ This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
8	EC	Excessive collision. This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC configuration register, this bit is set after first collision and the transmission of the packet is aborted.
9	LC	Late collision. This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including preamble in MII mode). This bit is not valid if underflow error is set.
10	NC	No carrier.

Table continues on the next page...

Table 659. TDES3 normal descriptor (write-back format) (continued)

Bit	Symbol	Description
		This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
11	LOC	Loss of carrier. This bit indicates that loss of carrier occurred during packet transmission. This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
12	PCE	Payload checksum error. This bit indicates that the checksum offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either because of insufficient bytes, as indicated by the payload length field of the IP header, or the MTL starting to forward the packet to the MAC transmitter in store and forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store and forward mode.
13	FF	Packet flushed. This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
14	JT	Jabber timeout. This bit indicates that the MAC transmitter has experienced a jabber time-out. This bit is set only when the JD bit of the MAC configuration register is not set.
15	ES	Error summary. This bit indicates the logical OR of the following bits: • TDES3[0]: IP header error • TDES3[14]: jabber timeout • TDES3[13]: packet flush • TDES3[12]: payload checksum error • TDES3[11]: loss of carrier • TDES3[10]: no carrier • TDES3[9]: late collision • TDES3[8]: excessive collision • TDES3[3]: excessive deferral • TDES3[2]: underflow error
16	-	Reserved.
17	TTSS	Tx timestamp status. This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the transmit packet. This field is valid only when the last segment control bit (TDES3[28]) in a descriptor is set.
27:18	-	Reserved.

Table continues on the next page...

Table 659. TDES3 normal descriptor (write-back format) (continued)

Bit	Symbol	Description
28	LD	Last descriptor. This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
29		First descriptor This bit indicates that the buffer contains the first segment of a packet.
30	CTXT	Context type. This bit should be set to 0 for normal descriptor.
31	OWN	Own bit. When this bit is set, it indicates that the Ethernet DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.

70.3.10.4 Receive descriptor

The DMA in the Ethernet module attempts to read a descriptor only if the tail pointer is different from the base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC. Otherwise, the performance of the DMA is impacted greatly because of the unavailability of the descriptors. In such situations, the Rx FIFO in MTL becomes full and starts dropping packets.

All RX descriptors are prepared by the software and given to the DMA as normal descriptors with the content as shown in receive normal descriptor (read format). The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the Rx DMA will close the descriptor with the corresponding packet status. The format of this status is given in the receive normal descriptor (write-back format).

In the receive descriptor (read format), if the buffer address field is all 0s, the Ethernet block does not transfer data to that buffer and skips to the next buffer or next descriptor.

In the receive descriptor (read format), if the buffer address field is all 0s, Ethernet does not transfer data to that buffer and skips to the next buffer or next descriptor.

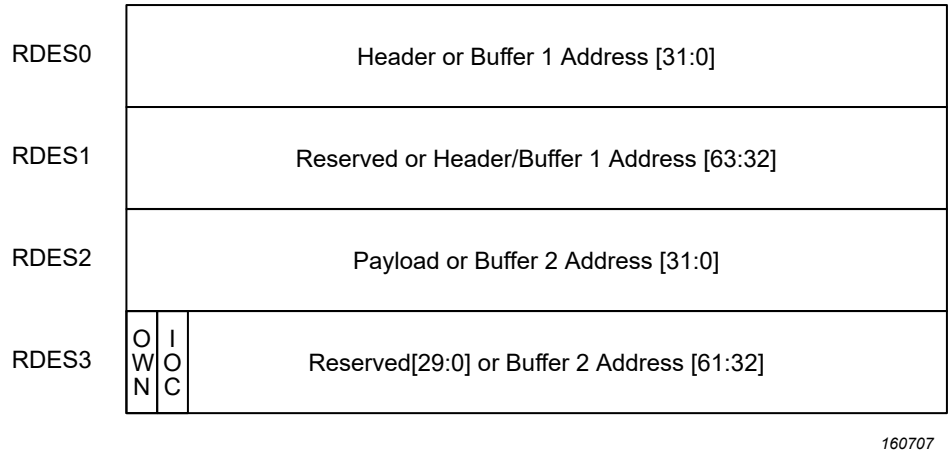


Figure 427. Receive normal descriptor (read format)

70.3.10.4.1 Receive normal descriptor (read format)

Table 660. RDES0 normal descriptor (read format)

Bit	Symbol	Description
31:0	BUF1AP	Header or buffer 1 address pointer. These bits indicate the physical address of buffer 1. The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a Write operation with RDES0[1:0] (or RDES0[2:0]/[3:0] in case of 64-/128-bit configuration) as zero. However, the packet data is shifted as per actual offset as given by buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

Table 661. RDES1 normal descriptor (read format)

Bit	Symbol	Description
31:0	-	Reserved.

Table 662. RDES2 normal descriptor (read format)

Bit	Symbol	Description
31:0	BUF2AP	Buffer 2 address pointer. These bits indicate the physical address of buffer 2. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores BUF2AP[3:0 or 2:0 or 1:0] (corresponding to 128- or 64- or 32-bit data-bus) and writes to the complete location.

Table 663. RDES3 normal descriptor (read format)

Bit	Symbol	Description
23:0	-	Reserved.
24	BUF1V	Buffer 1 address valid. When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by buffer 1 address in RDES1 can be used by the DMA to write received packet data.
25	BUF2V	Buffer 1 address valid. When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by buffer 1 address in RDES1 can be used by the DMA to write received packet data.
29:26	-	Reserved.
30	IOC	Interrupt enabled on completion When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.

Table continues on the next page...

Table 663. RDES3 normal descriptor (read format) (continued)

Bit	Symbol	Description
31	OWN	Own bit. When this bit is set, it indicates that the Ethernet module's DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception. • The buffers associated with the descriptor are full.

70.3.10.4.2 Receive normal descriptor (write-back format)

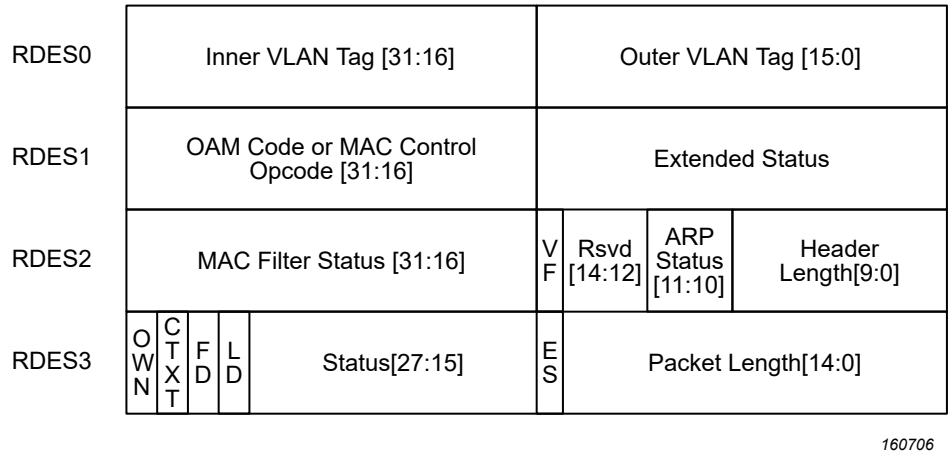


Figure 428. Receive descriptor write-back format

70.3.10.4.3 RDES0 normal descriptor (write-back format)

Table 664. RDES0 normal descriptor (write-back format)

Bit	Symbol	Description
15:0	OVT	Outer VLAN Tag This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.
31:16	IVT	Inner VLAN Tag This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.

70.3.10.4.4 RDES1 normal descriptor (write-back format)

The status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

Table 665. RDES1 normal descriptor (write-back format)

Bit	Symbol	Description
2:0	PT	Payload type. These bits indicate the type of payload encapsulated in the IP datagram processed by the receive checksum offload engine (COE): • 0x0: Unknown type or IP/AV payload not processed • 0x1: UDP • 0x2: TCP • 0x3: ICMP • 0x4: IGMP if IPV4 header present bit is set else Reserved • 0x5: AV untagged control packet • 0x6: AV tagged data packet • 0x7: AV tagged control packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 0x0.
3	IPHE	IP header error. When this bit is set, it indicates either of the following: • The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. • The IP datagram version is not consistent with the Ethernet type value. • Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.
4	IPV4	IPV4 header present. This bit indicates that an IPV4 header is detected.
5	IPV6	IPv6 header present. This bit indicates that an IPV6 header is detected.
6	IPCB	IP checksum bypassed. This bit indicates that the checksum offload engine is bypassed.
7	IPCE	IP payload error. When this bit is set, it indicates either of the following: • The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. • The TCP, UDP, or ICMP segment length does not match the payload length value in the IP header field. • The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.
11:8	PMT	PTP message type. These bits are encoded to give the type of the message received:

Table continues on the next page...

Table 665. RDES1 normal descriptor (write-back format) (continued)

Bit	Symbol	Description
		• 0x0: No PTP message received • 0x1: SYNC (all clock types) • 0x2: Follow_Up (all clock types) • 0x3: Delay Req (all clock types) • 0x4: Delay Resp (all clock types) • 0x5: Pdelay Req (in peer-to-peer transparent clock) • 0x6: Pdelay Resp (in peer-to-peer transparent clock) • 0x7: Pdelay Resp follow-up (in peer-to-peer transparent clock) • 0x8: Announce • 0x9: Management • 0xA: Signaling • 0xF: PTP packet with Reserved message type • Others: Reserved
12	PFT	PTP packet type. This bit indicates that the PTP message is sent directly over Ethernet.
13	PV	PTP version. This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format.
14	TSA	Timestamp available. When timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the last descriptor bit (RDES3[28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
15	TD	Timestamp dropped. This bit indicates that the timestamp was captured for this packet but it got dropped in the MTL Rx FIFO because of overflow
31:16	OPC	OAM sub-type code, or MAC control packet opcode OAM sub-type code. If bits 18:16 of RDES3 are set to 0x7, this field contains the OAM sub-type and code fields. MAC control packet opcode If bits 18:16 of RDES3 are set to 0x6, this field contains the MAC control packet opcode field.

70.3.10.4.5 RDES2 normal descriptor (write-back format)

Table 666. RDES2 normal descriptor (write-back format)

Bit	Symbol	Description
15:0	-	Reserved.
16	SAF	SA address filter fail. When this bit is set, it indicates that the packet failed the SA filter in the MAC.
17	DAF	Destination address filter fail. When this bit is set, it indicates that the packet failed the DA filter in the MAC.

Table continues on the next page...

Table 666. RDES2 normal descriptor (write-back format) (continued)

Bit	Symbol	Description
18	-	Reserved. Always 0.
26:19	MADRM	MAC address match. When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset.
31:20	--	Reserved.

70.3.10.4.6 RDES3 normal descriptor (write-back format)

Table 667. RDES3 normal descriptor (write-back format)

Bit	Symbol	Description
14:0	PL	Packet length. These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and either the descriptor error (RDES3[13]) or overflow error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the last descriptor and error summary bits are not set, this field indicates the accumulated number of bytes that are transferred for the current packet.
15	ES	Error summary. When this bit is set, it indicates the logical OR of the following bits: • RDES3[24]: CRC error • RDES3[19]: dribble error • RDES3[20]: receive error • RDES3[22]: watchdog timeout • RDES3[21]: overflow error • RDES3[23]: giant packet This field is valid only when the LD bit of RDES3 is set.
18:16	LT	Length/type field. This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: • 0x0: The packet is a length packet • 0x1: The packet is a type packet. • 0x3: The packet is a ARP request packet type • 0x4: The packet is a type packet with VLAN tag • 0x5: The packet is a type packet with double VLAN tag

Table continues on the next page...

Table 667. RDES3 normal descriptor (write-back format) (continued)

Bit	Symbol	Description
		• 0x6: The packet is a MAC control packet type • 0x7: The packet is a OAM packet type • 0x2: Reserved
19	DE	Dribble bit error. When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII mode.
20	RE	Receive error. When this bit is set, it indicates that the gmii_rxer_i signal is asserted while the gmii_rxdv_i signal is asserted during packet reception. This error also includes carrier extension error in the MII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
21	OE	Overflow error. When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store and forward mode, all partial packets are dropped completely in Rx FIFO.
22	RWT	Receive watchdog timeout. When this bit is set, it indicates that the receive watchdog timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
23	GP	Giant packet. When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9,018 or 9,022 bytes if jumbo packet enable is set). Giant packet indicates only the packet length. It does not cause any packet truncation.
24	CE	CRC error. When this bit is set, it indicates that a cyclic redundancy check (CRC) error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
25	RS0V	Receive status RDES0 valid. When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	Receive status RDES1 valid. When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
27	RS2V	Receive status RDES2 valid. When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.

Table continues on the next page...

Table 667. RDES3 normal descriptor (write-back format) (continued)

Bit	Symbol	Description
28	LD	Last descriptor. When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
29	FD	First descriptor. When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet.
30	CTXT	Receive context descriptor. When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0x0 to this bit for normal receive descriptor.
31	OWN	Own bit. When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full

70.3.11 Initialization Information

This section provides the instructions for initializing the DMA or MAC registers in the proper sequence.It contains the following sections:

- Initializing DMA
- Initializing MTL registers
- Initializing MAC

70.3.11.1 Initializing DMA

Follow these steps to initialize the DMA controller:

1. Assert a software reset by setting bit 0 of the DMA mode register. This resets all of the MAC internal registers and logic.
2. Wait for the completion of the reset process (poll bit 0 of the DMA mode register, which is only cleared after the reset operation is completed).
3. Program the following fields to initialize the DMA SYSBUS mode register :
   - AAL
   - Fixed burst or undefined burst
   - Burst mode values for AHB bus interface.
4. Create a descriptor list for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (set bit 31 of descriptor TDES3/RDES3).
5. Program the transmit length registers DMA_CH0_TXDESC_RING_LENGTH, DMA_CH1_TXDESC_RING_LENGTH, DMA_CH0_RX_CONTROL2 and DMA_CH1_RX_CONTROL2. The ring length programmed must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor DMA_CH0_TXDESC_LIST_ADDR, DMA_CH1_TXDESC_LIST_ADDR, DMA_CH0_RXDESC_LIST_ADDR,

and DMA_CH1_RXDESC_LIST_ADDR. Also, program transmit and receive tail pointer registers indicating to the DMA about the available descriptors DMA_CH0_TXDESC_TAIL_POINTER, DMA_CH1_TXDESC_TAIL_POINTER and DMA_CH0_RXDESC_TAIL_POINTER, DMA_CH1_RXDESC_TAIL_POINTER.

7. Program the settings of the following registers for the parameters like maximum burst-length (PBL) initiated by DMA, descriptor skip lengths, OSF in case of TxDMA, RBSZ in case of RxDMA. DMA_CH0_CTRL, DMA_CH1_CTRL DMA_CH0_TX_CTRL, DMA_CH1_TX_CTRL, DMA_CH0_RX_CTRL, DMA_CH1_RX_CTRL.
8. Enable the interrupts by programming the DMA_CH0_INTERRUPT_ENABLE, DMA_CH1_INTERRUPT_ENABLE registers.
9. Start the receive and transmit DMAs by setting SR (bit 0) of the DMA_CH0_RX_CTRL, DMA_CH1_RX_CTRL and ST (bit 0) of the DMA_CH0_RX_CTRL, DMA_CH1_RX_CTRL register.

70.3.11.2 Initializing MTL

Complete the following steps to initialize the MTL registers:

1. Program the Tx scheduling (SCHALG) and receive arbitration algorithm (RAA) fields in MTL operation mode register to initialize the MTL operation in case of multiple Tx and Rx Queues.
2. Program the Rx Queue to DMA mapping in MTL Rx Queue and DMA channel mapping register.
3. Program the following fields to initialize the mode of operation in the MTL TxQ0 operation mode register
   - Transmit store and forward (TSF) or transmit threshold control (TTC) in case of threshold mode
   - Transmit Queue enable (TXQEN) to value 0x2 to enable Transmit Queue 0
   - Transmit Queue Size (TQS).
4. Program the following fields to initialize the mode of operation in the MTL RxQ0 operation mode register
   - Receive store and forward (RSF) or RTC in case of threshold mode
   - Flow control activation and deactivation thresholds for MTL Receive FIFO (RFA and RFD).
   - Error packet and undersized good packet forwarding enable (FEP and FUP)
   - Rx Queue size (RQS).
5. Repeat steps 3 and 4 for MTL Tx and Rx Queue 1.

70.3.11.3 Initializing MAC

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is completed before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: MAC_Address0_High and MAC_Address0_Low. If more than one MAC address is enabled in your configuration, program the MAC addresses appropriately.
2. Program the following fields to set the appropriate filters for the incoming frames in the MAC frame filter register
   - Receive all
   - Promiscuous mode
   - Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the MAC transmit flow control register
   - Pause time and other pause frame control bits
   - Transmit flow control bits.
   - Flow control busy.
4. Program the MAC_Interrupt_Enable register as required, and if applicable, for your configuration.

5. Program the appropriate fields in the MAC_Configuration register, for example, Inter-packet gap while transmission and jabber disabled.
6. Set bit 0 and 1 in the MAC configuration registers to start the MAC transmitter and receiver.

70.3.11.3.1 Host bus burst access

The DMA attempts to execute fixed-length burst transfers on the AHB master interface if configured to do so (FB bit of DMA register 0). The maximum burst length is indicated and limited by the PBL field (DMA register 0[13:8]). The receive and transmit descriptors are always accessed in the maximum possible (limited by PBL or 16 x 8/bus width) burst-size for the 16-bytes to be read.

The transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise (no fixed-length burst), it transfers data using INCR (undefined length) and SINGLE transactions.

The receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in MTL receive FIFO or when the end of frame (when it is less than the configured burst-length) is detected in the receive FIFO. The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the AHB interface, then dummy transfers are performed in order to complete the fixed-burst. Otherwise (FB bit of DMA_SYS_BUS_MODE register is reset), it transfers data using INCR (undefined length) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the AHB initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16), because the AHB interface does not support more than INCR16.

70.3.11.3.2 Host data buffer alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example: buffer read

If the transmit buffer address is 0x0000 0FF2 (for 32-bit data bus), and 15 bytes need to be transferred, then the DMA reads five full words from address 0x00000FF0, but when transferring data to the MTL transmit FIFO, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures it transfers a full 32-bit data to the MTL transmit FIFO, unless it is the end-of-frame.

Example: buffer write

If the receive buffer address is 0x0000 0FF2 (for 64-bit data bus) and 16 bytes of a received frame need to be transferred, then the DMA writes 3 full words from address 0x0000 0FF0. But the first 2 bytes of first transfer and the last 6 bytes of the third transfer have dummy data.

70.3.11.3.3 Buffer size calculations

The DMA does not update the size fields in the transmit and receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES1) towards the MAC core. If a descriptor is marked as first (FS bit of TDES1 is set), then the DMA marks the first transfer from the buffer as the start of a frame. If a descriptor is marked as last (LS bit of TDES1), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0[29:16]) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The receive DMA always transfers the start of the next frame with a new descriptor.

Even when the start address of a receive buffer is not aligned to the system bus's data width, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the receive descriptor to have a 0x1002 offset. The receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual usable space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes, because of the start address offset.

70.3.11.3.4 DMA arbiter

The arbiter inside the DMA module performs the arbitration between the transmit and receive channel accesses to the AHB master interface. Two types of arbitrations are possible: round-robin, and fixed-priority.

When round-robin arbitration is selected (DA bit of DMA_MODE register) is also set, then the transmit DMA gets priority over the receive DMA.

70.3.12 Application Information

This section provides the instructions for initializing the DMA or MAC registers in the proper sequence.It contains the following sections:

- Performing normal receive and transmit operation
- Stopping and starting transmission
- Programming guidelines for multi-channel multi-queuing
- Programming guidelines for MII link state transitions
- Programming guidelines for IEEE 1588 timestamping
- Programming guidelines for AV feature
- Programming guidelines for energy-efficient Ethernet

70.3.12.1 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the host (either transmit or receive).
2. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and writing the descriptor tail pointer to Tx/Rx tail pointer register (DMA_CH0_TXDESC_TAIL_PTR / DMA_CH1_TXDESC_TAIL_PTR and DMA_CH0_RXDESC_TAIL_PTR DMA_CH1_RXDESC_TAIL_PTR). Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process DMA current host transmit register Table 854 and DMA current host receive register.
4. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process (register DMA current host transmit buffer address register and DMA current host receive buffer address register.

70.3.12.2 Stopping and Starting Transmission

Complete the following steps to pause the transmission for some time. The steps are provided for channel 0. The similar step can be used for other channels.

1. Disable the transmit DMA by clearing bit 0 (ST) of The DMA channel 0 transmit control Table 843 and The DMA channel 1 transmit control registers.
2. Wait for any previous frame transmissions to complete. It can be checked by reading the appropriate bits of the MTL TxQ0 debug register (TRCSTS is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing bit (RE) and bit 1(TE) of the MAC configuration register.
4. Disable the receive DMA, after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of MTL TxQ0 debug register, PRXQ=0 and RXQSTS=00).
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in MTL TxQ0 debug register, and RXQSTS is 0 in MTL RxQ0 debug register
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

NOTE

Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. The Software should change these parameters only when the MAC transmitter and receiver are not active. Similarly the DMA related configuration should not be changed when transmit and receive DMA are in active.

70.3.12.3 Multi-channel multi-queuing

70.3.12.3.1 Transmit

Complete the following steps for transmit.

1. Program the Tx Queue size in the TQS field of MTL TxQ0 operation mode register if queue 0 is used or MTL TxQ1 operation mode register if queue1 is used. Based on the value programmed in TQS field, the size of the queue is determined. In the transmit operation, the number of channels is equal to the number of queues. Due to this reason, the channel to queue mapping is fixed.
2. For a queue to be used, the queue needs to be enabled in TXQEN in the corresponding MTL TxQ operation mode register (MTL TxQ0 operation mode register or MTL TxQ1 operation mode register).
3. The scheduling method needs to be programmed in SCHALG of the MTL operation mode register.
4. Program the corresponding quantum weight register as per the selected algorithm. In case of CBS algorithm in AVB queues, the MTL_TxQ1_ETS_Control, MTL_TxQ1_SendSlopeCredit, MTL_TxQ1_HiCredit, and MTL_TxQ1_LoCredit registers also need to be programmed as required. These registers are not available for Q0.

70.3.12.3.2 Receive

Complete the following steps for transmit.

1. Program the Receive queue size in the RQS field of MTL_RxQ[n]_Operation_Mode Register. Based on the value programmed in RQS field, the size of the queue is determined.
2. Enable both the Rx Queues 0 and 1 in the fields RXQ0EN and RXQ1EN in the MAC Rx Queue control0 register for AV.
3. The MAC routes the Rx packets to the Rx Queues based on the following packet types:
4. AV PTP Packets: Based on the programming of AVPTPQ in MAC Rx Queue control register.
5. AV untagged control packets: Based on the programming of AVCPQ in MAC Rx Queue control 1 register.
6. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping

7. Program the RAA field of MTL operation mode register to select the arbitration algorithm to decide which RxQ is read out from the Rx FIFO memory.
8. Program the MTL RxQ control register to decide the weights and the packet arbitration for each RxQ.
9. Set corresponding DADMACH bit in MTL Rx Queue and DMA channel mapping register to select dynamic mapping of packets in each Rx Queue.
10. In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of the DCS field in the lowest MAC address register.

1. Program the Receive queue size in the RQS field of MTL_RxQ[n]_Operation_Mode Register. Based on the value programmed in the RQS field, the size of the queue is determined.
2. Enable the Receive Queues 0 to 4 in the fields RXQ0EN to RXQ4EN in MAC_RxQ_Ctrl0 Register for AV or DCB. In DMA configurations, SR bit of statically or dynamically mapped DMA_CH[n]_Rx_Control Register and corresponding RXQ[n]_EN in MAC_RxQ_Ctrl0 Register needs to be enabled.
3. The MAC routes the Rx packets to the Rx Queues based on the following packet types:
    a. AV PTP Packets: Based on the programming of AVPTPQ in MAC_RxQ_Ctrl1 Register.
    b. AV Untagged Control packets: Based on the programming of AVCPQ in MAC_RxQ_Ctrl1 Register.
    c. Data Center Bridging (DCB) related Link Layer Discovery Protocol (LLDP) packets. Program DCBCPQ in MAC_RxQ_Ctrl1 Register to indicate to MAC which queue should get the DCB packets.
    d. VLAN Tag Priority field in VLAN Tagged packets: Program PSRQ4-0 of the MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 Register for the routing of tagged packets based on the USP (user Priority) field of the received packets to the Rx Queues 0 to 4.
    e. The AV tagged control and data packets are also routed based on PSRQ field of the MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 registers.

NOTE

The priorities set in PSRQ4-0 should be unique.

4. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping:
    a. Program the RAA field of MTL_Operation_Mode register to select the arbitration algorithm to decide which RxQ is read out from the RxFIFO memory.
    b. Program the MTL_RxQ[n]_Control to decide the weights and the packet arbitration for each RxQ.
    c. If static mapping is programmed in MTL_RxQ_DMA_Map[n] register (RXQ[n]DADMACH is reset to 0), bits RXQx2DMA and others need to be programmed to select the channel for which each queue is mapped.
    d. Set RXQ[n]DADMACH bit in MTL_RxQ_DMA_Map0 Register to select dynamic mapping of packets in each RxQueue.
    e. In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of the DCS field in the lowest MAC Address Register.

70.3.12.4 MII link state transitions

70.3.12.4.1 Transmit and receive clocks are running when the link is down

1. Disable the transmit DMA (if applicable) by clearing bit 0 (ST) of the DMA channel control registers (both CH0 and CH1). or
2. Disable the MAC receiver by clearing bit 0(RE) of the MAC configuration register.
3. Wait for any previous frame transmissions to complete. It can be checked this by reading the appropriate bits of MTL_TXQ0_DBG register (TRCSTS is not 01). or Flush the Tx FIFO for faster empty operation.

4. Disable the MAC transmitter by clearing bit 1(TE) of the MAC configuration register.
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in (MTL_TXQ0_DBG register and RXQSTS is 0 in MTL_RxQ0_DBG register).
6. After the link is up, read the PHY registers to know the latest configuration and accordingly program the MAC registers.
7. Restart the operation by starting the Tx DMA, and then enabling the MAC transmitter and receiver. The Rx DMA not need to be disabled. As the receiver is disabled, the FIFO does not get any data in the Rx FIFO.

Complete the following steps when the link is down but the transmit and receive clocks are running: The steps are provided for channel 0.

70.3.12.4.2 Transmit and receive clocks are stopped when the link is down

Complete the following steps when the link is down but the transmit and receive clocks are stopped: The steps are provided for channel 0.

1. Disable the MAC transmitter and receiver by clearing bits RE and TE of MAC configuration register. This will not take effect immediately as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait for the completion of the transfer of any partial frame if any at time of stopping of transmit/receive clock. This can be checked by reading the MAC debug register (should be all zero).Some old packets may still remain in the Tx FIFO as the MAC transmitter is stopped.
4. Read the PHY registers to know the latest operating mode and accordingly program the MAC registers.
5. Restart the MAC transmitter and receiver by setting RE and TE bits.

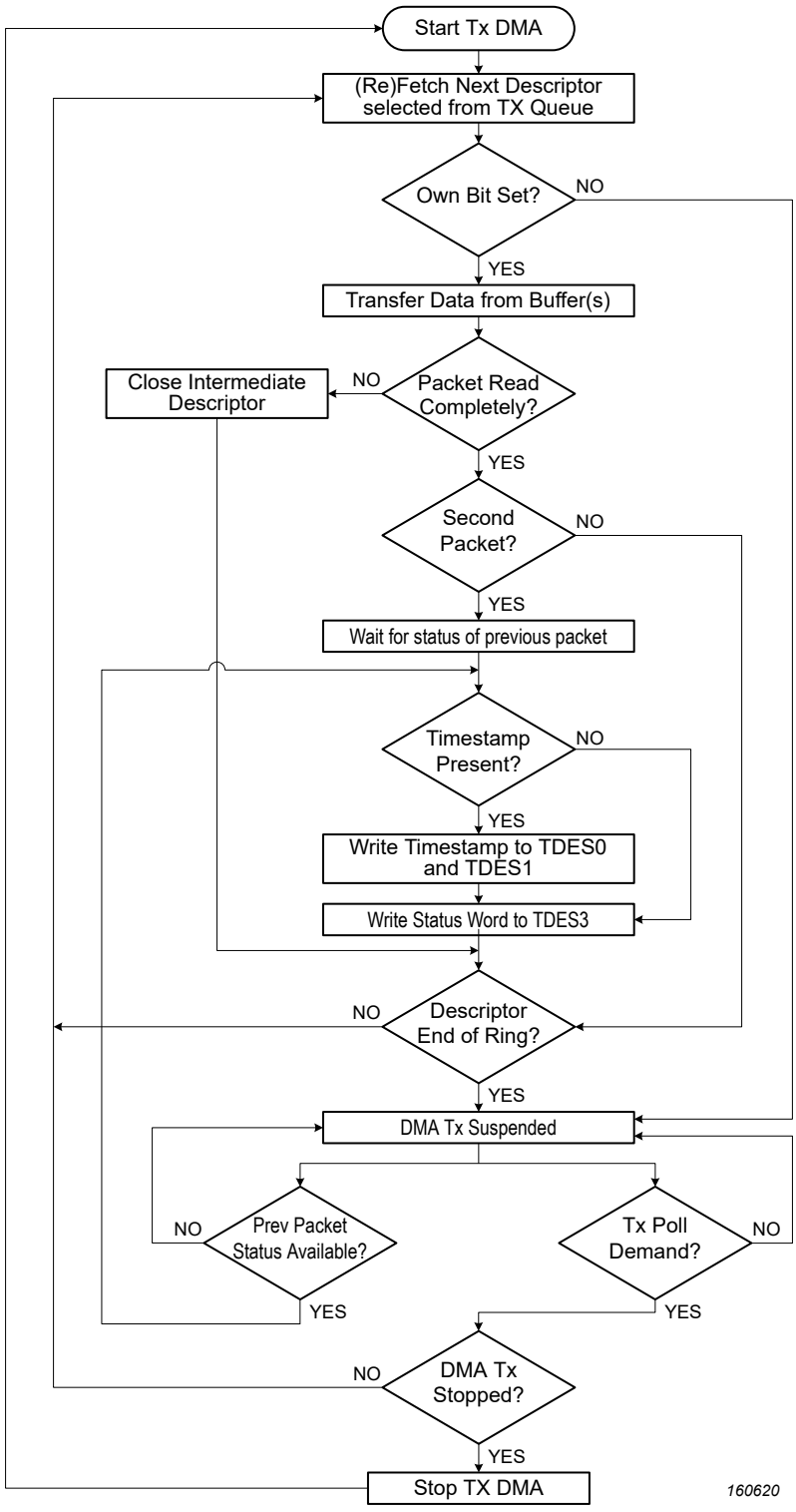


Figure 429. Transmit DMA operation in OSF mode

70.3.12.5 IEEE 1588 timestamping

70.3.12.5.1 Initialization guideline for system time generation

The timestamp feature can be enabled by setting bit 0 of the MAC timestamp control register. However, the timestamp counter must be initialized after this bit is set. Complete the following steps during Ethernet initialization:

1. Mask the timestamp trigger interrupt by clearing the bit 16 of MAC interrupt enable register.
2. Set bit 0 of the MAC timestamp control register to enable timestamping.
3. Program MAC sub-second increment register based on the PTP clock frequency.
4. If a fine correction approach is used, program MAC timestamp addend register and set bit 5 of MAC timestamp control register.
5. Poll the MAC timestamp control register until bit 5 is cleared.
6. Program bit 1 of MAC timestamp control register to select the fine update method (if required).
7. Program MAC system time seconds update register and MAC system time nanoseconds update register with the appropriate time value.
8. Set bit 2 in the MAC timestamp control register. The timestamp counter starts operation as soon as it is initialized with the value written in the timestamp update registers.
9. Enable the MAC receiver and transmitter for proper timestamping.

If timestamp operation is disabled by clearing bit 0 of the MAC timestamp control register, you need to repeat all these steps to restart the timestamp operation.

70.3.12.5.2 System time correction

To synchronize or update the system time in one process (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (MAC system time seconds update register and MAC system time nanoseconds update register.
2. Set bit 3 (TSUPDT) of the MAC timestamp control register. The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm explained in, calculate the rate by which you want to make the system time increments slower or faster.
2. Update the MAC timestamp addend register with the new value and set bit 5 of the MAC timestamp control register.
3. Wait for the time for which you want the new value of the addend register to be active. You can do this by enabling the timestamp trigger interrupt after the system time reaches the target value.
4. Enable the timestamp interrupt in bit 12 of MAC interrupt enable register.
5. Set bit 4 in MAC timestamp control register.
6. When this trigger causes an interrupt, read MAC interrupt status register.
7. Reprogram MAC timestamp addend register with the old value and set bit 5 again.

70.3.12.6 AV feature

70.3.12.6.1 Initializing the DMA

Complete the following steps to initialize the DMA:

1. Assert a software reset by setting bit 0 in DMA mode register. This resets all of the MAC internal registers and logic (bit-0 of DMA_Mode).

2. Wait for the completion of the reset process (poll bit 0 of the DMA_Mode, which is only cleared after the reset operation is completed).
3. Program the following fields to initialize the DMA_SYSBUS_MODE register:
    a. AAL
    b. Fixed burst or undefined burst
    c. Burst mode values in case of AHB bus interface, OSR_LMT in case of AXI bus interface.
    d. If fixed length value is enabled, select the maximum burst length possible on the AXI Bus (bits [7:1])
4. Create a descriptor list for transmit and receive. In addition, ensure that the descriptors are owned by DMA (set bit 31 of descriptor TDES3/RDES3). When OSF mode is used, at least two TX descriptors are required. For more information about descriptors, see Descriptor structure.
5. Make sure that the driver software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the Transmit and Receive Ring length registers (DMA_CH(#i)_TXDESC_RING_LENGTH (for i = 0; i <= 1) and DMA_CH(#i)_RX_CONTROL2 (for i = 0; i <= 1)). The ring length programmed must be at least 4.

NOTE

The descriptor address from the start to the end of the ring must not cross the 4GB boundary.

7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA_CH(#i)_TxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1), DMA_CH(#i)_RxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)). Also, program transmit and receive tail pointer registers indicating to the DMA about the available descriptors (DMA_CH(#i)_TxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1))

NOTE

For 40-bit or 48-bit addressing mode, program the higher address List registers (DMA_CH[n]_TxDesc_List_HAddress, DMA_CH[n]_RxDesc_List_HAddress).

The tailpointer registers must be advanced to the location immediately after the descriptors that are set, for the DMA to know that additional descriptors are available.

8. Program the following fields to initialize the mode of operation in the MTL_TxQ0 operation mode register.
    - Transmit store and forward (TSF)
    - Transmit threshold control (TTC)
    - Transmit Queue enable (TXQEN) to value 0x2 to enable Transmit Queue 0
    - Transmit Queue size (TQS)
9. Enable the interrupts by programming the DMA_CH0_INTERRUPT_ENABLE and register.
10. Start the Receive and Transmit DMAs by setting SR (bit 0) of the DMA_CH(#i)_RX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1) and ST (bit 0) of the DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register.
11. Repeat steps 4 to 9 for all the Tx DMA and Rx DMA channels selected in the hardware.

70.3.12.6.2 Enabling slot number checking

The slot number check feature can be used to specify the intervals at which the DMA channels mapped to AV queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the Ethernet memory. Complete the following steps to enable the slot number checking:

1. Enable timestamping by following the steps described in "Initialization Guideline for System Time Generation".

2. Make sure that the SLOTNUM field (bits 22:19) of TDES3 Normal Descriptor (Read Format) contains a valid slot number. You can read the current reference slot number from the DMA_CH(#i)_Slot_Function_Control_Status (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1).
3. Set Bit 0 (ESC) of the Slot Function Control and Status register of a channel to enable the slot number checking.

70.3.12.6.3 Enabling average bits per slot reporting

The MTL TxQ ETS status register of a particular AV channel provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

1. of a channel to generate the average bits per slot interrupt.
2. of a channel on each interrupt.

The frequency of this interrupt depends on the value programmed in step 2. For example, when you program value 0 in the SLC field, the interrupt is generated every 125 microseconds.

When not required, you can disable this interrupt to stop the interrupt flooding. The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled. When high, bit 1 (ABPSIS) of the MTL TxQ ETS status register indicates that a new value is updated in the ABS field.

70.3.12.7 Energy efficient Ethernet

70.3.12.7.1 Entering and exiting the Tx LPI mode

Complete the following steps during Ethernet core initialization:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX CLK stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode.)
3. Program Bits[25:16] and Bits[15:0] in MAC_LPI_TIMERS_CONTROL register.
4. Read the link status of the PHY chip by using the MDIO interface and update bit 17 of the MAC LPI control status register accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program the MAC_1US_TIC_COUNTER as per the frequency of the clock used for accessing the CSR slave port.
6. Program the MAC_LPI_ENTRY_TIMER register (LPIET) with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bit[20:19]) of MAC_LPI_CONTROL_STATUS register to enable the auto-entry into LPI and auto-exit of MAC from LPI state.
8. Program the MAC_1US_TIC_COUNTER as per the frequency of the clock used for accessing the CSR slave port.
9. Program the MAC_LPI_ENTRY_TIMER register (LPIET) with the IDLE time for which the MAC should wait before entering the LPI state on its own.
10. Set LPITE and LPITXA (bit[20:19]) of MAC_LPI_CONTROL_STATUS register to enable the auto-entry into LPI and auto-exit of MAC from LPI state.
11. Set Bit 16 of MAC_LPI_CONTROL_STATUS register to make the MAC Transmitter enter the LPI state. The MAC enters the LPI mode after completing all scheduled packets and remains IDLE for the time indicated by LPIET. It sets the TLPIEN (bit[0]) after entry to LPI state before step 12.
12. When a packet is scheduled for transmission (when the TxDMA comes out of IDLE state or when a packet is presented at ATI or MTI interface), the MAC Transmitter exits LPI state automatically. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.

13. MAC Transmitter re-enters LPI state if it remains IDLE for LPIET time and sets the TLPIEN bit and the entry-exit cycle continues.
14. MAC Transmitter re-enters LPI state if it remains IDLE for LPIET time and sets the TLPIEN bit and the entry-exit cycle continues.
15. Reset LPITXEN in case the application wants to over-ride the auto-entry/exit modes and make the MAC Transmitter exit the LPI state directly.
16. Start the receive and transmit DMA by setting bit 0 of the DMA channel 0, channel 1 transmit control register and bit 0 of DMA channel 0, channel 1 receive control register.

NOTE

To ensure MAC enter the LPI state only after completion of transmission of all the queued frames in the Tx FIFO, you should set Bit 19 in MAC_LPI_Control_Status register.

To switch off the GMII Transmit Clock during the LPI state, use the sbd_tx_clk_gating_ctrl_o signal for gating the clock input.

70.3.12.7.2 Enabling Slot Number Checking

The slot number check feature can be used to specify the intervals at which the DMA channels mapped to AV queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the Ethernet memory. Complete the following steps to enable the slot number checking:

1. Enable timestamping by following the steps described in Initialization guideline for system time generation.
2. Make sure that the SLOTNUM field (bits 22:19) of the TDES3 normal descriptor (read format) contains a valid slot number. The current reference slot number can be read from the DMA channel 0 and channel 1 slot function control register
3. Set bit 0 (ESC) of the slot function control and status register of a channel to enable the slot number checking

70.4 Signals

NOTE

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

Table 668. Ethernet Signals

Pin function	Direction	Description
MIIM interface		
MDIO	I/O	Ethernet MIIM data input and output.
MDC	O	MIIM clock.
RMII interface		
RXD[1:0]	I	Receive data.
TXD[1:0]	O	Transmit data.
RX_DV	I	Receive data valid.
TX_EN	O	Ttransmit data enable.

Table continues on the next page...

Table 668. Ethernet Signals (continued)

Pin function	Direction	Description
MII interface		
RXD[3:0]	I	Receive data.
TXD[3:0]	O	Transmit data.
COL	I	Collision detect.
CRS	I	Carrier sense.
TX_ER	O	Transmit error.
TX_CLK	I	Transmit clock.
RX_CLK	I	Receive clock.
RX_ER	I	Receive error.
TX_EN	O	Transmit enable.

70.5 Memory Map and register definition

NOTE

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

This section includes the eNET module memory map and detailed descriptions of all registers.

NOTE

After a write operation, when a register content is being transferred to a different clock domain, user needs to wait at least for four cycles of the destination clock (PHY Receive clock, PHY Transmit clock, or PTP clock) before writing to the same register.

70.5.1 ENET register descriptions

70.5.1.1 ENET memory map

ENET0 base address: 4010_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	MAC Configuration (MAC_CONFIGURATION)	32	RW	0000_8000h
4h	MAC Extended Configuration Register (MAC_EXT_CONFIGURATION)	32	RW	0000_0000h
8h	MAC Packet Filter (MAC_PACKET_FILTER)	32	RW	0000_0000h
Ch	Watchdog Timeout (MAC_WATCHDOG_TIMEOUT)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
50h	MAC VLAN Tag Control (MAC_VLAN_TAG_CTRL)	32	RW	0000_0000h
60h	VLAN Tag Inclusion or Replacement (MAC_VLAN_INCL)	32	RW	0000_0000h
64h	MAC Inner VLAN Tag Inclusion or Replacement (MAC_INNER_VLAN_INCL)	32	RW	0000_0000h
70h	MAC Q0 Tx Flow Control (MAC_Q0_TX_FLOW_CTRL)	32	RW	0000_0000h
90h	MAC Rx Flow Control (MAC_RX_FLOW_CTRL)	32	RW	0000_0000h
94h	Receive Queue Control 4 (MAC_RXQ_CTRL4)	32	RW	0000_0000h
A0h	Receive Queue Control 0 (MAC_RXQ_CTRL0)	32	RW	0000_0000h
A4h	Receive Queue Control 1 (MAC_RXQ_CTRL1)	32	RW	0000_0000h
A8h	Receive Queue Control 2 (MAC_RXQ_CTRL2)	32	RW	0000_0000h
B0h	Interrupt Status (MAC_INTERRUPT_STATUS)	32	R	0000_0000h
B4h	Interrupt Enable (MAC_INTERRUPT_ENABLE)	32	RW	0000_0000h
B8h	Receive Transmit Status (MAC_RX_TX_STATUS)	32	R	0000_0000h
C0h	PMT Control and Status (MAC_PMT_CONTROL_STATUS)	32	RW	0000_0000h
C4h	Remote Wakeup Filter (MAC_RWK_PACKET_FILTER)	32	RW	0000_0000h
D0h	LPI Control and Status (MAC_LPI_CONTROL_STATUS)	32	RW	0000_0000h
D4h	LPI Timers Control (MAC_LPI_TIMERS_CONTROL)	32	RW	03E8_0000h
D8h	Tx LPI Entry Timer Control (MAC_LPI_ENTRY_TIMER)	32	RW	0000_0000h
DCh	One-microsecond Reference Timer (MAC_ONEUS_TIC_COUNTER)	32	RW	0000_0063h
110h	MAC Version (MAC_VERSION)	32	R	0000_1052h
114h	MAC Debug (MAC_DEBUG)	32	R	0000_0000h
11Ch	Hardware Features 0 (MAC_HW_FEATURE0)	32	R	0E01_70E5h
120h	Hardware Features 1 (MAC_HW_FEATURE1)	32	R	0010_0104h
124h	Hardware Features 2 (MAC_HW_FEATURE2)	32	R	0004_1041h
128h	Hardware Features 3 (MAC_HW_FEATURE3)	32	R	0000_0230h
200h	MDIO Address (MAC_MDIO_ADDRESS)	32	RW	0000_0000h
204h	MAC MDIO Data (MAC_MDIO_DATA)	32	RW	0000_0000h
230h	CSR Software Control (MAC_CSR_SW_CTRL)	32	RW	0000_0000h
300h	MAC Address0 High (MAC_ADDRESS0_HIGH)	32	RW	8000_FFFFh
304h	MAC Address0 Low (MAC_ADDRESS0_LOW)	32	RW	FFFF_FFFFh
A70h	Indirect Access Control (INDIR_ACCESS_CTRL)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
A74h	Indirect Access Data (INDIR_ACCESS_DATA)	32	RW	0000_0000h
B00h	Timestamp Control (MAC_TIMESTAMP_CONTROL)	32	RW	0000_2000h
B04h	Subsecond Increment (MAC_SUB_SECOND_INCREMENT)	32	RW	0000_0000h
B08h	System Time Seconds (MAC_SYSTEM_TIME_SECONDS)	32	R	0000_0000h
B0Ch	System Time Nanoseconds (MAC_SYSTEM_TIME_NANOSECONDS)	32	R	0000_0000h
B10h	System Time Seconds Update (MAC_SYSTEM_TIME_SECONDS_UPDATE)	32	RW	0000_0000h
B14h	System Time Nanoseconds Update (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)	32	RW	0000_0000h
B18h	Timestamp Addend (MAC_TIMESTAMP_ADDEND)	32	RW	0000_0000h
B20h	Timestamp Status (MAC_TIMESTAMP_STATUS)	32	R	0000_0000h
B30h	Transmit Timestamp Status Nanoseconds (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)	32	R	0000_0000h
B34h	Transmit Timestamp Status Seconds (MAC_TX_TIMESTAMP_STATUS_SECONDS)	32	R	0000_0000h
B58h	Timestamp Ingress Correction Nanosecond (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)	32	RW	0000_0000h
B5Ch	Timestamp Egress Correction Nanosecond (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)	32	RW	0000_0000h
B68h	Timestamp Ingress Latency (MAC_TIMESTAMP_INGRESS_LATENCY)	32	R	0000_0000h
B6Ch	Timestamp Egress Latency (MAC_TIMESTAMP_EGRESS_LATENCY)	32	R	0000_0000h
B70h	PPS Control (MAC_PPS_CONTROL)	32	RW	0000_0000h
B80h	PPS0 Target Time Seconds (PPS0_TARGET_TIME_SECONDS)	32	RW	0000_0000h
B84h	PPS0 Target Time Nanoseconds (PPS0_TARGET_TIME_NANOSECONDS)	32	RW	0000_0000h
C00h	MTL Operation Mode (MTL_OPERATION_MODE)	32	RW	0000_0000h
C20h	MTL Interrupt Status (MTL_INTERRUPT_STATUS)	32	R	0000_0000h
C30h	Receive Queue and DMA Channel Mapping 0 (MTL_RXQ_DMA_MAP0)	32	RW	0000_0000h
D00h	Queue 0 Transmit Operation Mode (MTL_TXQ0_OPERATION_MODE)	32	RW	0000_0000h
D04h	Queue 0 Underflow Counter (MTL_TXQ0_UNDERFLOW)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
D08h	Queue 0 Transmit Debug (MTL_TXQ0_DEBUG)	32	R	0000_0000h
D14h	Queue 0 ETS Status (MTL_TXQ0_ETS_STATUS)	32	R	0000_0000h
D18h	Queue 0 Quantum or Weights (MTL_TXQ0_QUANTUM_WEIGHT)	32	RW	0000_0000h
D2Ch	Queue 0 Interrupt Control Status (MTL_Q0_INTERRUPT_CONTROL_STATUS)	32	RW	0000_0000h
D30h	Queue 0 Receive Operation Mode (MTL_RXQ0_OPERATION_MODE)	32	RW	0000_0000h
D34h	Queue 0 Missed Packet and Overflow Counter (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)	32	R	0000_0000h
D38h	Queue 0 Receive Debug (MTL_RXQ0_DEBUG)	32	R	0000_0000h
D3Ch	Queue 0 Receive Control (MTL_RXQ0_CONTROL)	32	RW	0000_0000h
D40h	Queue 1 Transmit Operation Mode (MTL_TXQ1_OPERATION_MODE)	32	RW	0000_0000h
D44h	Queue 1 Underflow Counter (MTL_TXQ1_UNDERFLOW)	32	R	0000_0000h
D48h	Queue 1 Transmit Debug (MTL_TXQ1_DEBUG)	32	R	0000_0000h
D50h	Queue 1 ETS Control (MTL_TXQ1_ETS_CONTROL)	32	RW	0000_0000h
D54h	Queue 1 ETS Status (MTL_TXQ1_ETS_STATUS)	32	R	0000_0000h
D58h	Queue 1 idleSlopeCredit, Quantum or Weights (MTL_TXQ1_QUANTUM_WEIGHT)	32	RW	0000_0000h
D5Ch	Queue 1 sendSlopeCredit (MTL_TXQ1_SENDSLOPECREDIT)	32	RW	0000_0000h
D60h	Queue 1 hiCredit (MTL_TXQ1_HICREDIT)	32	RW	0000_0000h
D64h	Queue 1 loCredit (MTL_TXQ1_LOCREDIT)	32	RW	0000_0000h
D6Ch	Queue 1 Interrupt Control Status (MTL_Q1_INTERRUPT_CONTROL_STATUS)	32	RW	0000_0000h
D70h	Queue 1 Receive Operation Mode (MTL_RXQ1_OPERATION_MODE)	32	RW	0000_0000h
D74h	Queue 1 Missed Packet and Overflow Counter (MTL_RXQ1_MISSED_PACKET_OVERFLOW_CNT)	32	R	0000_0000h
D78h	Queue 1 Receive Debug (MTL_RXQ1_DEBUG)	32	R	0000_0000h
D7Ch	Queue 1 Receive Control (MTL_RXQ1_CONTROL)	32	RW	0000_0000h
1000h	DMA Bus Mode (DMA_MODE)	32	RW	0000_0000h
1004h	DMA System Bus Mode (DMA_SYSBUS_MODE)	32	RW	0000_0000h
1008h	DMA Interrupt Status (DMA_INTERRUPT_STATUS)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
100Ch	DMA Debug Status 0 (DMA_DEBUG_STATUS0)	32	R	0000_0000h
1100h	DMA Channel 0 Control (DMA_CH0_CONTROL)	32	RW	0000_0000h
1104h	DMA Channel 0 Transmit Control (DMA_CH0_TX_CONTROL)	32	RW	0000_0000h
1108h	DMA Channel 0 Receive Control (DMA_CH0_RX_CONTROL)	32	RW	0000_0000h
1114h	Channel 0 Tx Descriptor List Address register (DMA_CH0_TXDESC_LIST_ADDRESS)	32	RW	0000_0000h
111Ch	Channel 0 Rx Descriptor List Address register (DMA_CH0_RXDESC_LIST_ADDRESS)	32	RW	0000_0000h
1120h	Channel 0 Tx Descriptor Tail Pointer (DMA_CH0_TXDESC_TAIL_POINTER)	32	RW	0000_0000h
1128h	Channel 0 Rx Descriptor Tail Pointer (DMA_CH0_RXDESC_TAIL_POINTER)	32	RW	0000_0000h
112Ch	Channel 0 Tx Descriptor Ring Length (DMA_CH0_TXDESC_RING_LENGTH)	32	RW	0000_0000h
1130h	Channeli Receive Control (DMA_CH0_RX_CONTROL2)	32	RW	0000_0000h
1134h	Channeli Interrupt Enable (DMA_CH0_INTERRUPT_ENABLE)	32	RW	0000_0000h
1138h	Channel 0 Receive Interrupt Watchdog Timer (DMA_CH0_RX_INTERRUPT_WATCHDOG_TIMER)	32	RW	0000_0000h
113Ch	Channel 0 Slot Function Control and Status (DMA_CH0_SLOT_FUNCTION_CONTROL_STATUS)	32	RW	0000_07C0h
1144h	Channel 0 Current Application Transmit Descriptor (DMA_CH0_CURRENT_APP_TXDESC)	32	R	0000_0000h
114Ch	Channel 0 Current Application Receive Descriptor (DMA_CH0_CURRENT_APP_RXDESC)	32	R	0000_0000h
1154h	Channel 0 Current Application Transmit Buffer Address (DMA_CH0_CURRENT_APP_TXBUFFER)	32	R	0000_0000h
115Ch	Channel 0 Current Application Receive Buffer Address (DMA_CH0_CURRENT_APP_RXBUFFER)	32	R	0000_0000h
1160h	DMA Channel 0 Status (DMA_CH0_STATUS)	32	RW	0000_0000h
1164h	Channel 0 Missed Frame Counter (DMA_CH0_MISS_FRAME_CNT)	32	R	0000_0000h
116Ch	Channel 0 Receive ERI Counter (DMA_CH0_RX_ERI_CNT)	32	R	0000_0000h
1180h	DMA Channel 1 Control (DMA_CH1_CONTROL)	32	RW	0000_0000h
1184h	DMA Channel 1 Transmit Control (DMA_CH1_TX_CONTROL)	32	RW	0000_0000h
1188h	DMA Channel 1 Receive Control (DMA_CH1_RX_CONTROL)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
1194h	Channel 1 Tx Descriptor List Address (DMA_CH1_TXDESC_LIST_ADDRESS)	32	RW	0000_0000h
119Ch	Channel 1 Rx Descriptor List Address (DMA_CH1_RXDESC_LIST_ADDRESS)	32	RW	0000_0000h
11A0h	Channel 1 Tx Descriptor Tail Pointer (DMA_CH1_TXDESC_TAIL_POINTER)	32	RW	0000_0000h
11A8h	Channel 1 Rx Descriptor Tail Pointer (DMA_CH1_RXDESC_TAIL_POINTER)	32	RW	0000_0000h
11ACh	Channel 1 Tx Descriptor Ring Length (DMA_CH1_TXDESC_RING_LENGTH)	32	RW	0000_0000h
11B0h	DMA Channel 1 Receive Control (DMA_CH1_RX_CONTROL2)	32	RW	0000_0000h
11B4h	Channel 1 Interrupt Enable (DMA_CH1_INTERRUPT_ENABLE)	32	RW	0000_0000h
11B8h	Channel 1 Receive Interrupt Watchdog Timer (DMA_CH1_RX_INTERRUPT_WATCHDOG_TIMER)	32	RW	0000_0000h
11BCh	Channel 1 Slot Function Control and Status (DMA_CH1_SLOT_FUNCTION_CONTROL_STATUS)	32	RW	0000_07C0h
11C4h	Channel 1 Current Application Transmit Descriptor (DMA_CH1_CURRENT_APP_TXDESC)	32	R	0000_0000h
11CCh	Channel 1 Current Application Receive Descriptor (DMA_CH1_CURRENT_APP_RXDESC)	32	R	0000_0000h
11D4h	Channel 1 Current Application Transmit Buffer Address (DMA_CH1_CURRENT_APP_TXBUFFER)	32	R	0000_0000h
11DCh	Channel 1 Current Application Receive Buffer Address (DMA_CH1_CURRENT_APP_RXBUFFER)	32	R	0000_0000h
11E0h	DMA Channel 1 Status (DMA_CH1_STATUS)	32	RW	0000_0000h
11E4h	Channel 1 Missed Frame Counter (DMA_CH1_MISS_FRAME_CNT)	32	R	0000_0000h
11ECh	Channel 1 Receive ERI Counter (DMA_CH1_RX_ERI_CNT)	32	R	0000_0000h

70.5.1.2 MAC Configuration (MAC_CONFIGURATION)

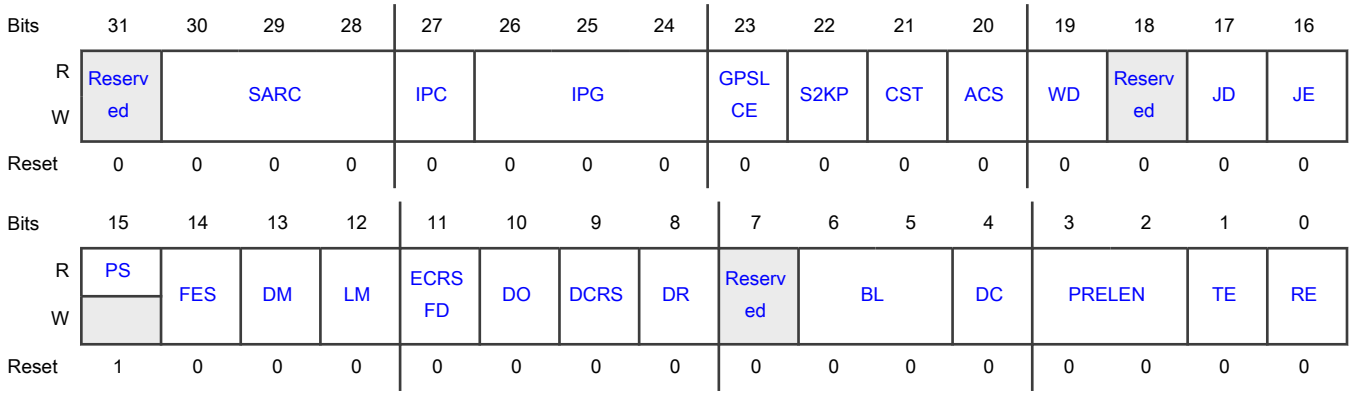
Offset

Register	Offset
MAC_CONFIGURATION	0h

Function

The MAC Configuration Register establishes the operating mode of the MAC.

Diagram



Fields

Field	Function
31 —	Reserved
30-28 SARC	Source Address Insertion or Replacement Control This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]: 2'b0x: The mti_sa_ctrl_i and ati_sa_ctrl_i input signals control the SA field generation. 2'b10: If Bit 30 is set to 0, the MAC inserts the content of the MAC Address 0 registers in the SA field of all transmitted packets. If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected while configuring the IP, the MAC inserts the content of the MAC Address 1 registers in the SA field of all transmitted packets. 2'b11: If Bit 30 is set to 0, the MAC replaces the content of the MAC Address 0 registers in the SA field of all transmitted packets. If Bit 30 is set to 1 and the MAC Address Register 1 is enabled, the MAC replaces the content of the MAC Address 1 registers in the SA field of all transmitted packets. Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. 000b - mti_sa_ctrl_i and ati_sa_ctrl_i input signals control the SA field generation 010b - Contents of MAC Addr-0 inserted in SA field 011b - Contents of MAC Addr-0 replaces SA field

Table continues on the next page...

Table continued from the previous page...

Field	Function
	110b - Contents of MAC Addr-1 inserted in SA field 111b - Contents of MAC Addr-1 replaces SA field
27 IPC	Checksum Offload When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled. The Layer 3 and Layer 4 Packet Filter and Enable Split Header features automatically selects the IPC Full Checksum Offload Engine on the Receive side. When any of these features are enabled, you must set the IPC bit. 0b - IP header/payload checksum checking is disabled 1b - IP header/payload checksum checking is enabled
26-24 IPG	Inter-Packet Gap These bits control the minimum IPG between packets during transmission. This range of minimum IPG is valid in full-duplex mode. In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG. This function (IPG less than 96 bit times) is valid only when MAC_EXT_CONFIGURATION[EIPGEN] bit is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in MAC_Ext_Configuration register. 000b - 96 bit times IPG 001b - 88 bit times IPG 010b - 80 bit times IPG 011b - 72 bit times IPG 100b - 64 bit times IPG 101b - 56 bit times IPG 110b - 48 bit times IPG 111b - 40 bit times IPG
23 GPSLCE	Giant Packet Size Limit Control Enable When this bit is set, the MAC considers the value in GPSL field in MAC_Ext_Configuration register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit. When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet). The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018

Table continues on the next page...

Table continued from the previous page...

Field	Function
	bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status. 0b - Giant Packet Size Limit Control is disabled 1b - Giant Packet Size Limit Control is enabled
22 S2KP	IEEE 802.3as Support for 2K Packets When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets. When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see the Table, Giant Packet Status based on S2KP and JE Bits. Note: When the JE bit is set, setting this bit has no effect on the giant packet status. 0b - Support upto 2K packet is disabled 1b - Support upto 2K packet is Enabled
21 CST	CRC stripping for Type packets When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. 0b - CRC stripping for Type packets is disabled 1b - CRC stripping for Type packets is enabled
20 ACS	Automatic Pad or CRC Stripping When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is reset, the MAC passes all incoming packets to the application, without any modification. 0b - Automatic Pad or CRC Stripping is disabled 1b - Automatic Pad or CRC Stripping is enabled
19 WD	Watchdog Disable When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes. When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes. 0b - Watchdog is enabled 1b - Watchdog is disabled
18	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
17 JD	Jabber Disable When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes. When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet. 0b - Jabber is enabled 1b - Jabber is disabled
16 JE	Jumbo Packet Enable When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status. 0b - Jumbo packet is disabled 1b - Jumbo packet is enabled
15 PS	Port Select This bit selects the Ethernet line speed. This bit, along with Bit 14, selects the exact line speed. In the 10/100 Mbps-only (always 1) or 1000 Mbps-only (always 0) configurations, this bit is read-only (RO) with appropriate value. In default 10/100/1000 Mbps configurations, this bit is read-write (R/W). 0b - For 1000 or 2500 Mbps operations 1b - For 10 or 100 Mbps operations
14 FES	Speed This bit selects the speed mode. 0b - 10 Mbps when PS bit is 1 and 1 Gbps when PS bit is 0 1b - 100 Mbps when PS bit is 1 and 2.5 Gbps when PS bit is 0
13 DM	Duplex Mode When this bit is set, the MAC operates in the full-duplex mode in which it can transmit and receive simultaneously. This bit is RO with default value of 1'b1 in the full-duplex-only configurations. 0b - Half-duplex mode 1b - Full-duplex mode
12 LM	Loopback Mode When this bit is set, the MAC operates in the loopback mode at GMII or MII. The (G)MII Rx clock input (clk_rx_i) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Loopback is disabled 1b - Loopback is enabled
11 ECRSFD	Enable Carrier Sense Before Transmission in Full-Duplex Mode When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low. When this bit is reset, the MAC transmitter ignores the status of the CRS signal. 0b - ECRSFD is disabled 1b - ECRSFD is enabled
10 DO	Disable Receive Own When this bit is set, the MAC disables the reception of packets when the gmii_txen_o is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY. This bit is not applicable in the full-duplex mode. 0b - Enable Receive Own 1b - Disable Receive Own
9 DCRS	Disable Carrier Sense During Transmission When this bit is set, the MAC transmitter ignores the (G)MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission. When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission. 0b - Enable Carrier Sense During Transmission 1b - Disable Carrier Sense During Transmission
8 DR	Disable Retry When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status. When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the half-duplex mode. 0b - Enable Retry 1b - Disable Retry
7 —	Reserved
6-5 BL	Back-Off Limit

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The back-off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000/2500 Mbps; 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. n = retransmission attempt. The random integer r takes the value in the range 0 <= r < 2^k This bit is applicable only in the half-duplex mode. 00b - k = min(n,10) 01b - k = min(n,8) 10b - k = min(n,4) 11b - k = min(n,1)
4 DC	Deferral Check When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288-bit times in 10 or 100 Mbps mode. If the MAC is configured for 1000/2500 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000-bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and a collision happens. Because of the collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode. 0b - Deferral check function is disabled 1b - Deferral check function is enabled
3-2 PRELEN	Preamble Length for Transmit packets These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode. 00b - 7 bytes of preamble 01b - 5 bytes of preamble 10b - 3 bytes of preamble 11b - Reserved
1 TE	Transmitter Enable When this bit is set, the Tx state machine of the MAC is enabled for transmission on the GMII or MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Transmitter is disabled 1b - Transmitter is enabled
0 RE	Receiver Enable When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the GMII or MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the GMII or MII interface. 0b - Receiver is disabled 1b - Receiver is enabled

70.5.1.3 MAC Extended Configuration Register (MAC_EXT_CONFIGURATION)

Offset

Register	Offset
MAC_EXT_CONFIGURATION	4h

Function

The MAC Extended Configuration Register establishes the operating mode of the MAC.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	Reserved	EIPG					EIPGEN	Reserved	Reserved			PDC	USP	SPEN	DCRCC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		GPSL													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
30 —	Reserved
29-25 EIPG	Extended Inter-Packet Gap The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits), along with IPG field in CONFIGURATION register, gives the minimum IPG greater than 96-bit times in steps of 8 bit times: {EIPG, IPG} 8'h00 - 104-bit times 8'h01 - 112-bit times 8'h02 - 120-bit times ---------- 8'hFF - 2144 bit times
24 EIPGEN	Extended Inter-Packet Gap Enable When this bit is set, the MAC interprets EIPG field and IPG field in CONFIGURATION register together as minimum IPG greater than 96-bit times in steps of 8-bit times. When this bit is reset, the MAC ignores EIPG field and interprets IPG field in CONFIGURATION register as minimum IPG less than or equal to 96-bit times in steps of 8-bit times. Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-Duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-Duplex mode. 0b - Extended Inter-Packet Gap is disabled 1b - Extended Inter-Packet Gap is enabled
23 —	Reserved
22-20 —	Reserved
19 PDC	Packet Duplication Control When this bit is set, the received packet with Multicast/Broadcast Destination address is routed to multiple Receive DMA Channels. The Receive DMA Channels is identified by the DCS field of MAC_ADDRESS(#i)_HIGH register corresponding to the MAC Address register that matches the Multicast/Broadcast Destination address in the received packet. The DCS field is interpreted to be a one-hot value, each bit corresponding to the Receive DMA Channel. When this bit is reset, the received packet is routed to single Receive DMA Channel. The Receive DMA Channel is identified by the DCS field of ADDRESS(#i)_High register corresponding to the MAC Address register that matches the Destination address in the received packet. The DCS field is interpreted as a binary value.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Packet Duplication Control is disabled 1b - Packet Duplication Control is enabled
18 USP	Unicast Slow Protocol Packet Detect When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the MAC_ADDRESS0_HIGH and MAC_ADDRESS0_LOW registers. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02). When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2015, Section 5. 0b - Unicast Slow Protocol Packet Detection is disabled 1b - Unicast Slow Protocol Packet Detection is enabled
17 SPEN	Slow Protocol Detection Enable When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid sub-types. When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets. 0b - Slow Protocol Detection is disabled 1b - Slow Protocol Detection is enabled
16 DCRCC	Disable CRC Checking for Received Packets When this bit is set, the MAC receiver does not check the CRC field in the received packets. When this bit is reset, the MAC receiver always checks the CRC field in the received packets. 0b - CRC Checking is enabled 1b - CRC Checking is disabled
15-14 —	Reserved
13-0 GPSL	Giant Packet Size Limit If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes. For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. When the Enable Double VLAN Processing option is selected, the MAC adds 8 bytes to the programmed value for double VLAN tagged packets. The value in this field is applicable when the GPSLCE bit is set in MAC_CONFIGURATION register.

70.5.1.4 MAC Packet Filter (MAC_PACKET_FILTER)

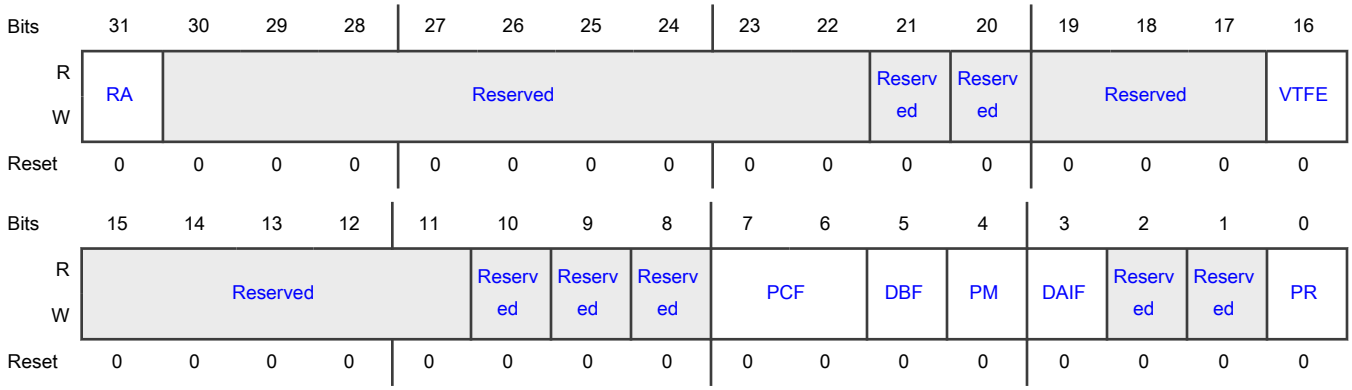
Offset

Register	Offset
MAC_PACKET_FILTER	8h

Function

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

Diagram



Fields

Field	Function
31 RA	Receive All When this bit is set, the MAC Receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word. When this bit is reset, the Receiver module passes only those packets to the application that pass the SA or DA address filter. 0b - Receive All is disabled 1b - Receive All is enabled
30-22 —	Reserved
21 —	Reserved
20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
19-17 —	Reserved
16 VTFE	VLAN Tag Filter Enable When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag. 0b - VLAN Tag Filter is disabled 1b - VLAN Tag Filter is enabled
15-11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7-6 PCF	Pass Control Packets These bits control the forwarding of all control packets (including unicast and multicast Pause packets). 00b - MAC filters all control packets from reaching the application 01b - MAC forwards all control packets except Pause packets to the application even if they fail the address filter 10b - MAC forwards all control packets to the application even if they fail the address filter 11b - MAC forwards the control packets that pass the Address filter
5 DBF	Disable Broadcast Packets When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast packets. 0b - Enable Broadcast Packets 1b - Disable Broadcast Packets
4 PM	Pass All Multicast

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit. 0b - Pass All Multicast is disabled 1b - Pass All Multicast is enabled
3 DAIF	DA Inverse Filtering When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed. 0b - DA Inverse Filtering is disabled 1b - DA Inverse Filtering is enabled
2 —	Reserved
1 —	Reserved
0 PR	Promiscuous Mode When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set. 0b - Promiscuous Mode is disabled 1b - Promiscuous Mode is enabled

70.5.1.5 Watchdog Timeout (MAC_WATCHDOG_TIMEOUT)

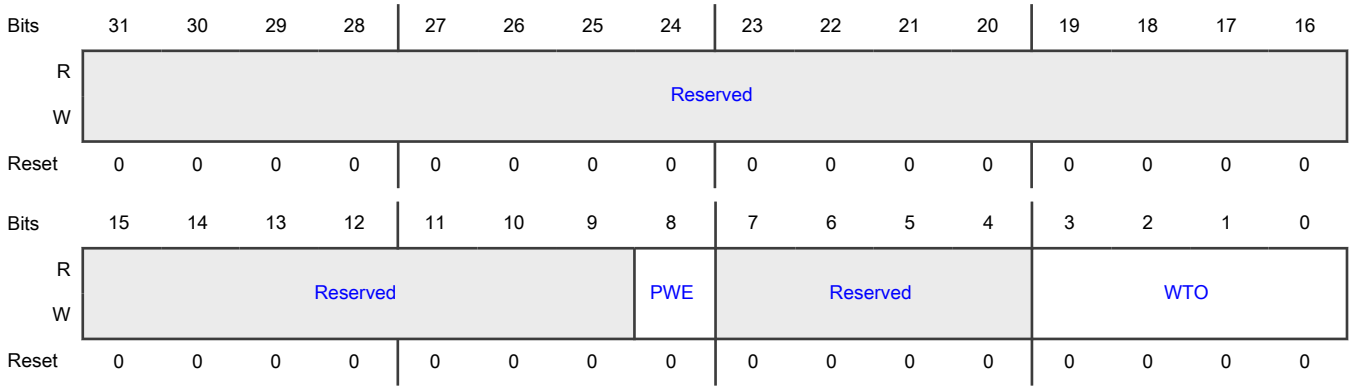
Offset

Register	Offset
MAC_WATCHDOG_TIMEOUT	Ch

Function

The Watchdog Timeout register controls the watchdog timeout for received packets.

Diagram



Fields

Field	Function
31-9 —	Reserved
8 PWE	Programmable Watchdog Enable When this bit is set and the WD bit of the MAC_CONFIGURATION register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in MAC_CONFIGURATION register. 0b - Programmable Watchdog is disabled 1b - Programmable Watchdog is enabled
7-4 —	Reserved
3-0 WTO	Watchdog Timeout When the PWE bit is set and the WD bit of the MAC_CONFIGURATION register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet. Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped. 0000b - 2 KB 0001b - 3 KB 0010b - 4 KB 0011b - 5 KB 0100b - 6 KB 0101b - 7 KB 0110b - 8 KB

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0111b - 9 KB 1000b - 10 KB 1001b - 11 KB 1010b - 12 KB 1011b - 13 KB 1100b - 14 KB 1101b - 15 KB 1110b - 16383 Bytes 1111b - Reserved

70.5.1.6 MAC VLAN Tag Control (MAC_VLAN_TAG_CTRL)

Offset

Register	Offset
MAC_VLAN_TAG_CTRL	50h

Function

This register is the redefined format of the MAC VLAN Tag Register. It is used for indirect addressing. It contains the address offset, command type and Busy Bit for CSR access of the Per VLAN Tag registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EIVLRXS	Reserved	EIVLS		ERIVLT	EDVLP	Reserved	EVLRXS	Reserved	EVLS		DOVLTC	ERSVLM	ESVL	VTIM	ETV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Enable Inner VLAN Tag in Rx Status

Table continues on the next page...

Table continued from the previous page...

Field	Function
EIVLRXS	Enable Inner VLAN Tag in Rx Status When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status. 0b - Inner VLAN Tag in Rx status is disabled 1b - Inner VLAN Tag in Rx status is enabled
30 —	Reserved
29-28 EIVLS	Enable Inner VLAN Tag Stripping on Receive 00b - Do not strip 01b - Strip if VLAN filter passes 10b - Strip if VLAN filter fails 11b - Always strip
27 ERIVLT	Enable Inner VLAN Tag Enable Inner VLAN Tag When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching. 0b - Inner VLAN tag is disabled 1b - Inner VLAN tag is enabled
26 EDVLP	Enable Double VLAN Processing When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present). 0b - Double VLAN Processing is disabled 1b - Double VLAN Processing is enabled
25 —	Reserved
24 EVLRXS	Enable VLAN Tag in Rx status Enable VLAN Tag in Rx status When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status. 0b - VLAN Tag in Rx status is disabled 1b - VLAN Tag in Rx status is enabled
23	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
22-21 EVLS	Enable VLAN Tag Stripping on Receive 00b - Do not strip 01b - Strip if VLAN filter passes 10b - Strip if VLAN filter fails 11b - Always strip
20 DOVLTC	Disable VLAN Type Check When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN. When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit. The VLAN filter is bypassed when VLAN Type of received packet do not match the programmed VLAN Type in the VLAN filter. 0b - VLAN Type Check is enabled 1b - VLAN Type Check is disabled
19 ERSVLM	Enable Receive S-VLAN Match When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets. The ERIVLT bit determines the VLAN tag position considered for filtering or matching. 0b - Receive S-VLAN Match is disabled 1b - Receive S-VLAN Match is enabled
18 ESVL	Enable S-VLAN When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets. 0b - S-VLAN is disabled 1b - S-VLAN is enabled
17 VTIM	VLAN Tag Inverse Match Enable When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched. 0b - VLAN Tag Inverse Match is disabled 1b - VLAN Tag Inverse Match is enabled
16 ETV	Enable 12-Bit VLAN Tag Comparison When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged

Table continues on the next page...

Table continued from the previous page...

Field	Function
	packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for hash-based VLAN filtering. When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN hash filtering. 0b - 12-bit VLAN Tag Comparison is disabled 1b - 12-bit VLAN Tag Comparison is enabled
15-0 VL	VLAN Tag Identifier for Receive Packets This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field: Bits[15:13]: User Priority Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) Bits[11:0]: VLAN Identifier (VID) field of VLAN tag When the ETV bit is set, only the VID is used for comparison. If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.

70.5.1.7 VLAN Tag Inclusion or Replacement (MAC_VLAN_INCL)

Offset

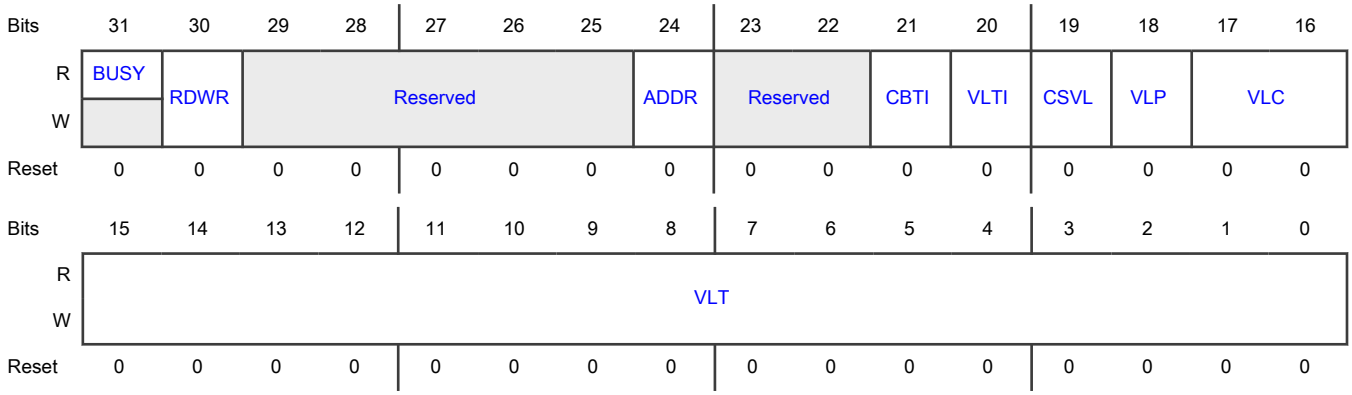
Register	Offset
MAC_VLAN_INCL	60h

Function

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit packets.

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit packets. It also contains the VLAN tag insertion controls.

Diagram



Fields

Field	Function
31 BUSY	Busy This bit indicates the status of the read/write operation of indirect access to the queue/channel specific VLAN inclusion register. For write operation write to a register is complete when this bit is reset. For read operation the read data is valid when the bit is reset. The application must make sure that this bit is reset before attempting subsequent access to this register. 0b - Busy status not detected 1b - Busy status detected
30 RDWR	Read write control This bit controls the read or write operation for indirectly accessing the queue/channel specific VLAN Inclusion register. When set indicates write operation and when reset indicates read operation. This does not have any effect when CBTI is reset. 0b - Read operation of indirect access 1b - Write operation of indirect access
29-25 —	Reserved
24 ADDR	Address Address This field selects one of the queue/channel specific VLAN Inclusion register for read/write access. This does not have any effect when CBTI is reset.
23-22 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 CBTI	Channel based tag insertion When this bit is set, outer VLAN tag is inserted for every packets transmitted by the MAC. The tag value is taken from the queue/channel specific VLAN tag register. The VLTI, VLP, VLC, and VLT fields of this register are ignored when this bit is set. When this bit is set, a write operation to byte 3 of this register initiates the read/write access to the indirect register. When reset, outer VLAN operation is based on the setting of VLTI, VLP, VLC and VLT fields of this register. 0b - Channel based tag insertion is disabled 1b - Channel based tag insertion is enabled
20 VLTI	VLAN Tag Input When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from: The Tx descriptor 0b - VLAN Tag Input is disabled 1b - VLAN Tag Input is enabled
19 CSVL	C-VLAN or S-VLAN When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets. 0b - C-VLAN type (0x8100) is inserted or replaced 1b - S-VLAN type (0x88A8) is inserted or replaced
18 VLP	VLAN Priority Control When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, the mti_vlan_ctrl_i control input is used and bits[17:16] are ignored. 0b - VLAN Priority Control is disabled 1b - VLAN Priority Control is enabled
17-16 VLC	VLAN Tag Control in Transmit Packets VLAN Tag Control in Transmit Packets 2'b00: No VLAN tag deletion, insertion, or replacement 2'b01: VLAN tag deletion The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags. 2'b10: VLAN tag insertion

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 2'b11: VLAN tag replacement The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8). Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. 00b - No VLAN tag deletion, insertion, or replacement 01b - VLAN tag deletion 10b - VLAN tag insertion 11b - VLAN tag replacement
15-0 VLT	VLAN Tag for Transmit Packets This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: Bits[15:13]: User priority Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

70.5.1.8 MAC Inner VLAN Tag Inclusion or Replacement (MAC_INNER_VLAN_INCL)

Offset

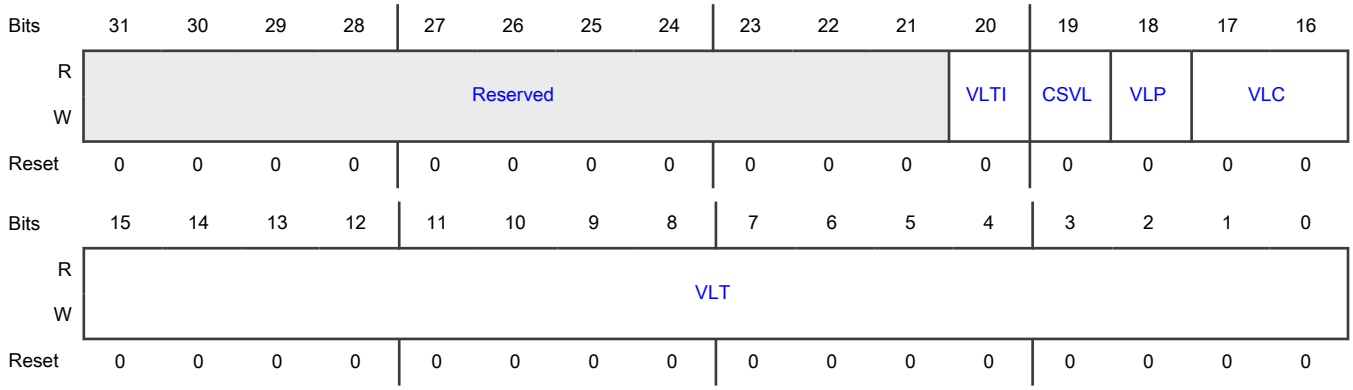
Register	Offset
MAC_INNER_VLAN_INCL	64h

Function

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the transmit packet.

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the transmit packet. It also contains the inner VLAN tag insertion controls.

Diagram



Fields

Field	Function
31-21 —	Reserved
20 VLTI	VLAN Tag Input When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor 0b - VLAN Tag Input is disabled 1b - VLAN Tag Input is enabled
19 CSVL	C-VLAN or S-VLAN C-VLAN or S-VLAN When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 17th and 18th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 17th and 18th bytes of transmitted packets. 0b - C-VLAN type (0x8100) is inserted 1b - S-VLAN type (0x88A8) is inserted
18 VLP	VLAN Priority Control VLAN Priority Control When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the mti_vlan_ctrl_i control input is used and the VLC field is ignored. 0b - VLAN Priority Control is disabled 1b - VLAN Priority Control is enabled
17-16 VLC	VLAN Tag Control in Transmit Packets VLAN Tag Control in Transmit Packets 2'b00: No VLAN tag deletion, insertion, or replacement

Table continues on the next page...

Table continued from the previous page...

Field	Function
	2'b01: VLAN tag deletion The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags. 2'b10: VLAN tag insertion The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 2'b11: VLAN tag replacement The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8). **NOTE** Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. 00b - No VLAN tag deletion, insertion, or replacement 01b - VLAN tag deletion 10b - VLAN tag insertion 11b - VLAN tag replacement
15-0 VLT	VLAN Tag for Transmit Packets This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: Bits[15:13]: User Priority Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

70.5.1.9 MAC Q0 Tx Flow Control (MAC_Q0_TX_FLOW_CTRL)

Offset

Register	Offset
MAC_Q0_TX_FLOW_CTRL	70h

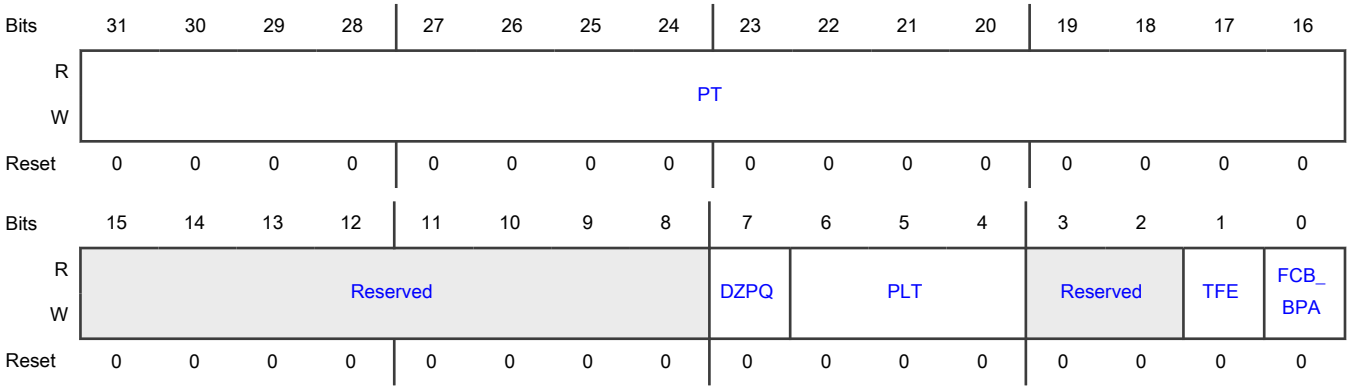
Function

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC.

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

When the PFCE bit in the MAC_Q0_RX_FLOW_CTRL register is enabled, this register controls the generation of the Priority Flow Control (PFC) frames with priorities mapped according to PSRQ0 in the MAC_Q0_RXQ_CTRL2 register.

Diagram



Fields

Field	Function
31-16 PT	Pause Time This field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the (G)MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.
15-8 —	Reserved
7 DZPQ	Disable Zero-Quanta Pause When this bit is set, it disables the automatic generation of the zero-quanta Pause packets on de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled. 0b - Zero-Quanta Pause packet generation is enabled 1b - Zero-Quanta Pause packet generation is disabled
6-4 PLT	Pause Low Threshold This field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values. The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times. 000b - Pause Time minus 4 Slot Times (PT -4 slot times) 001b - Pause Time minus 28 Slot Times (PT -28 slot times) 010b - Pause Time minus 36 Slot Times (PT -36 slot times) 011b - Pause Time minus 144 Slot Times (PT -144 slot times) 100b - Pause Time minus 256 Slot Times (PT -256 slot times) 101b - Pause Time minus 512 Slot Times (PT -512 slot times) 110b - Reserved
3-2 —	Reserved
1 TFE	Transmit Flow Control Enable Full-Duplex Mode: In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets. Half-Duplex Mode: In the half-duplex mode, when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled. 0b - Transmit Flow Control is disabled 1b - Transmit Flow Control is enabled
0 FCB_BPA	Flow Control Busy or Backpressure Activate This bit initiates a Pause packet in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. Full-Duplex Mode: In the full-duplex mode, this bit should be read as 1'b0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1'b1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 1'b0. You must not write to this register until this bit is cleared. Half-Duplex Mode:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When this bit is set (and TFE bit is set) in the half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically ORed with the mti_flowctrl_i input signal for the backpressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Flow Control Busy or Backpressure Activate is disabled 1b - Flow Control Busy or Backpressure Activate is enabled

70.5.1.10 MAC Rx Flow Control (MAC_RX_FLOW_CTRL)

Offset

Register	Offset
MAC_RX_FLOW_CTRL	90h

Function

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							Reserved	Reserved						UP	RFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-9 —	Reserved
8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-2 —	Reserved
1 UP	Unicast Pause Packet Detect A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in MAC_Address0_High and MAC_Address0_Low. When this bit is reset, the MAC only detects Pause packets with unique multicast address. Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011. 0b - Unicast Pause Packet Detect disabled 1b - Unicast Pause Packet Detect enabled
0 RFE	Receive Flow Control Enable When this bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in half-duplex mode, the decode function of the Pause packet is disabled. When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time. 0b - Receive Flow Control is disabled 1b - Receive Flow Control is enabled

70.5.1.11 Receive Queue Control 4 (MAC_RXQ_CTRL4)

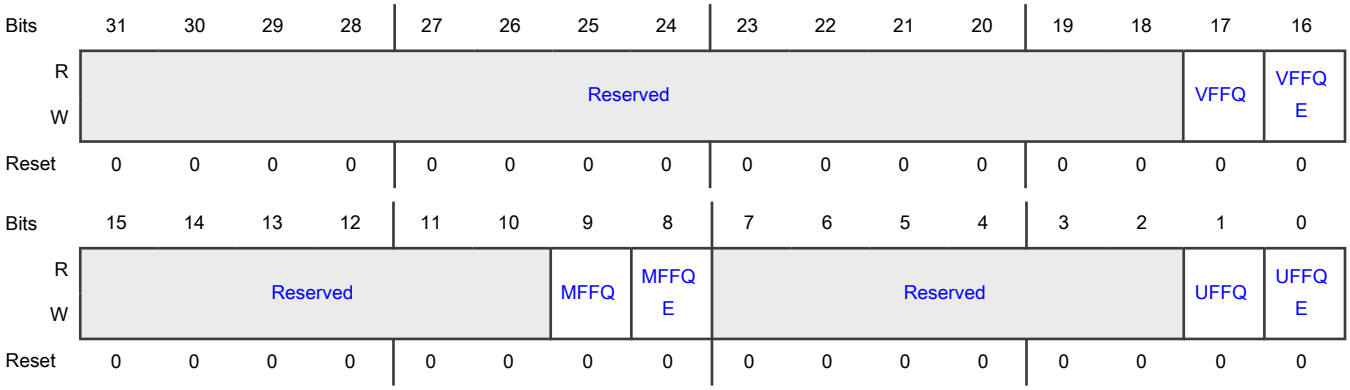
Offset

Register	Offset
MAC_RXQ_CTRL4	94h

Function

The Receive Queue Control 4 register controls the routing of unicast and multicast packets that fail the destination or source address filter to the Rx queues.

Diagram



Fields

Field	Function
31-18 —	Reserved
17 VFFQ	VLAN Tag Filter Fail Packets Queue This field holds the Rx queue number to which the tagged packets failing the destination or source address filter (and UFFQE/MFFQE not enabled) or failing the VLAN tag filter must be routed to. This field is valid only when the VFFQE bit is set.
16 VFFQE	VLAN Tag Filter Fail Packets Queuing Enable When this bit is set, the tagged packets which fail the destination or source address filter or fail the VLAN tag filter, are routed to the Rx Queue Number programmed in the VFFQ. When this bit is reset, the tagged packets which fail the destination or source address filter or fail the VLAN tag filter are routed based on other routing options. This bit is valid only when the RA bit of the MAC_Packet_Filter register is set. 0b - VLAN tag Filter Fail Packets Queuing is disabled 1b - VLAN tag Filter Fail Packets Queuing is enabled
15-10 —	Reserved
9 MFFQ	Multicast Address Filter Fail Packets Queue. This field holds the Rx queue number to which the Multicast packets failing the destination or source address filter are routed to. This field is valid only when the MFFQE bit is set.
8 MFFQE	Multicast Address Filter Fail Packets Queuing Enable. When this bit is set, the Multicast packets which fail the destination or source address filter is routed to the Rx Queue Number programmed in the MFFQ. When this bit is reset, the Multicast packets which fail the Destination or Source address filter is routed based on other routing options. This bit is valid only when the RA bit of the MAC_Packet_Filter register is set.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Multicast Address Filter Fail Packets Queuing is disabled 1b - Multicast Address Filter Fail Packets Queuing is enabled
7-2 —	Reserved
1 UFFQ	Unicast Address Filter Fail Packets Queue. This field holds the Rx queue number to which the Unicast packets failing the destination or source address filter are routed to. This field is valid only when the UFFQE bit is set.
0 UFFQE	Unicast Address Filter Fail Packets Queuing Enable. When this bit is set, the Unicast packets which fail the destination or source address filter is routed to the Rx Queue Number programmed in the UFFQ. When this bit is reset, the Unicast packets which fail the destination or source address filter is routed based on other routing options. This bit is valid only when the RA bit of the MAC_Packet_Filter register is set. 0b - Unicast Address Filter Fail Packets Queuing is disabled 1b - Unicast Address Filter Fail Packets Queuing is enabled

70.5.1.12 Receive Queue Control 0 (MAC_RXQ_CTRL0)

Offset

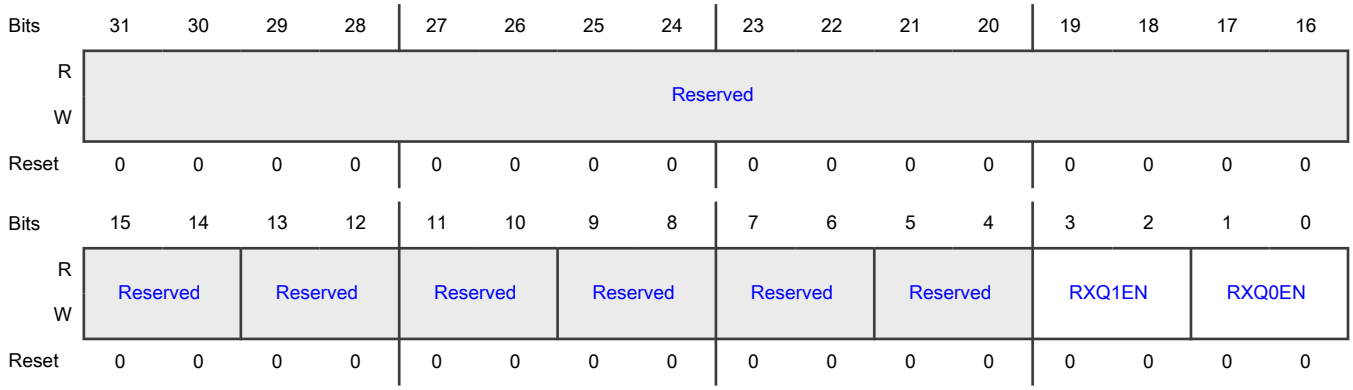
Register	Offset
MAC_RXQ_CTRL0	A0h

Function

The Receive Queue Control 0 register controls the queue management in the MAC Receiver.

Note: In multiple Rx queues configuration, all the queues are disabled by default. Enable the Rx queue by programming the corresponding field in this register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11-10 —	Reserved
9-8 —	Reserved
7-6 —	Reserved
5-4 —	Reserved
3-2 RXQ1EN	Receive Queue 1 Enable This field is similar to the RXQ0EN field. 00b - Queue not enabled 01b - Queue enabled for AV 10b - Queue enabled for DCB/Generic 11b - Reserved
1-0	Receive Queue 0 Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
RXQ0EN	This field indicates whether Rx Queue 0 is enabled for AV or DCB. 00b - Queue not enabled 01b - Queue enabled for AV 10b - Queue enabled for DCB/Generic 11b - Reserved

70.5.1.13 Receive Queue Control 1 (MAC_RXQ_CTRL1)

Offset

Register	Offset
MAC_RXQ_CTRL1	A4h

Function

The Receive Queue Control 1 register controls the routing of multicast, broadcast, AV, DCB, and untagged packets to the Rx queues.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		TBRQE	OMCBCQ	Reserved	Reserved			TPQC		TACPQE	MCBCQEN	Reserved	MCBCQ		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	UPQ			Reserved	Reserved			Reserved	PTPQ			Reserved	AVCPQ		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29 TBRQE	Type Field Based Rx Queuing Enable When this bit is set, it enables Type Field based Rx Queuing where the Type field of received packet is compared with programmed TYP field in MAC_TMRQ_Regs(#i) and if a match occurs the packet is routed to the corresponding TMRQ field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
28 OMCBCQ	OMCBCQ Over-riding MC-BC queue priority select 1: Priority of MCBCQ is reduced and the received packet is first routed to PTPQ, AVCPQ, DCBCPQ depending on packet type. 0: Received Multicast/Broadcast packet is routed to MCBCQ. 0b - overriding MCBCQ priority disabled 1b - overriding MCBCQ priority enabled
27 —	Reserved
26-24 —	Reserved
23-22 TPQC	Tagged PTP over Ethernet Packets Queuing Control. Tagged PTP over Ethernet Packets Queuing Control. This field controls the routing of the VLAN Tagged PTPoE packets. If DWC_EQOS_AV_ENABLE is selected in the configuration, the following programmable options are allowed. 2'b00: VLAN Tagged PTPoE packets are routed as generic VLAN Tagged packet (based on PSRQ for only non-AV enabled Rx Queues). 2'b01: VLAN Tagged PTPoE packets are routed to Rx Queue specified by PTPQ field (That Rx Queue can be enabled for AV or non-AV traffic). 2'b10: VLAN Tagged PTPoE packets are routed to only AV enabled Rx Queues based on PSRQ. 2'b11: Reserved If DWC_EQOS_AV_ENABLE is not selected in the configuration, the following programmable options are allowed. 1'b0: VLAN Tagged PTPoE packets are routed as generic VLAN Tagged packet (based on PSRQ for DCB/Generic enabled Rx Queues). 1'b1: VLAN Tagged PTPoE packets are routed to Rx Queues specified by PTPQ field.
21 TACPQE	Tagged AV Control Packets Queuing Enable. Tagged AV Control Packets Queuing Enable. When set, the MAC routes the received Tagged AV Control packets to the Rx queue specified by AVCPQ field. When reset, the MAC routes the received Tagged AV Control packets based on the tag priority matching the PSRQ fields in MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 registers. 0b - Tagged AV Control Packets Queuing is disabled 1b - Tagged AV Control Packets Queuing is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
20 MCBCQEN	Multicast and Broadcast Queue Enable This bit specifies that Multicast or Broadcast packets routing to the Rx Queue is enabled and the Multicast or Broadcast packets must be routed to Rx Queue specified in MCBCQ field. 0b - Multicast and Broadcast Queue is disabled 1b - Multicast and Broadcast Queue is enabled
19 —	Reserved
18-16 MCBCQ	Multicast and Broadcast Queue Multicast and Broadcast Queue This field specifies the Rx Queue onto which Multicast or Broadcast Packets are routed. Any Rx Queue enabled for Generic/DCB/AV traffic can be used to route the Multicast or Broadcast Packets. 000b - Receive Queue 0 001b - Receive Queue 1
15 —	Reserved
14-12 UPQ	Untagged Packet Queue This field indicates the Rx Queue to which Untagged Packets are to be routed. Any Rx Queue enabled for Generic/DCB/AV traffic can be used to route the Untagged Packets. 000b - Receive Queue 0 001b - Receive Queue 1
11 —	Reserved
10-8 —	Reserved
7 —	Reserved
6-4 PTPQ	PTP Packets Queue This field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed. When the AV8021ASMEN bit of MAC_Timestamp_Control register is set, only untagged PTP over Ethernet packets are routed on an Rx Queue. If the bit is not set, then based on programming of TPQC field, both tagged and untagged PTPoE packets can be routed to this Rx Queue.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000b - Receive Queue 0 001b - Receive Queue 1
3 —	Reserved
2-0 AVCPQ	AV Untagged Control Packets Queue AV Untagged Control Packets Queue This field specifies the Receive queue on which the received AV tagged and untagged control packets are routed. The AV tagged (when TACPQE bit is set) and untagged control packets are routed to Receive queue specified by this field. 000b - Receive Queue 0 001b - Receive Queue 1

70.5.1.14 Receive Queue Control 2 (MAC_RXQ_CTRL2)

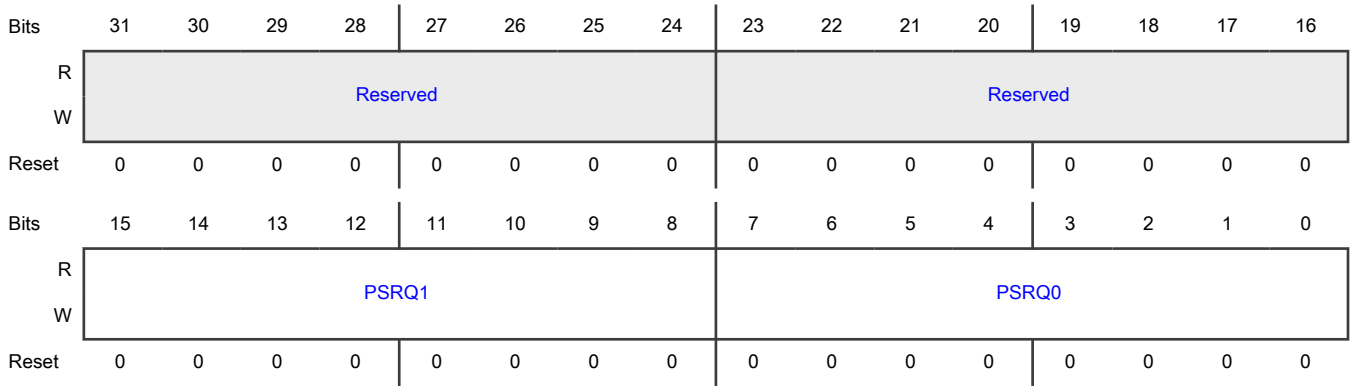
Offset

Register	Offset
MAC_RXQ_CTRL2	A8h

Function

This register controls the routing of tagged packets based on the USP (user Priority) field of the received packets to the RxQueues 0 to 3.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 —	Reserved
15-8 PSRQ1	Priorities Selected in the Receive Queue 1 This field decides the priorities assigned to Rx Queue 1. All packets with priorities that match the values set in this field are routed to Rx Queue 1. For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx Queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.
7-0 PSRQ0	Priorities Selected in the Receive Queue 0 This field decides the priorities assigned to Rx Queue 0. All packets with priorities that match the values set in this field are routed to Rx Queue 0. For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx Queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.

70.5.1.15 Interrupt Status (MAC_INTERRUPT_STATUS)

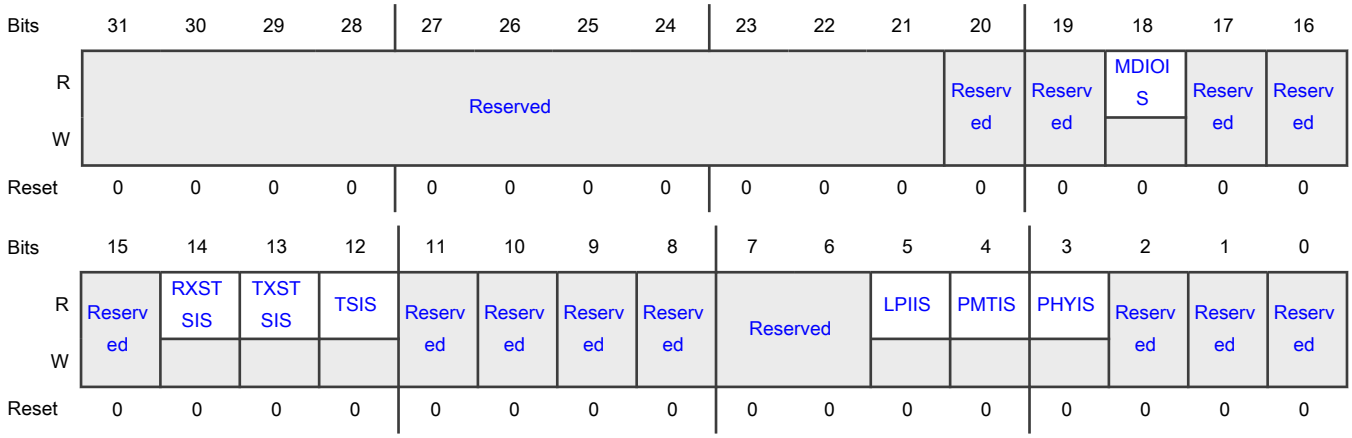
Offset

Register	Offset
MAC_INTERRUPT_STATUS	B0h

Function

The Interrupt Status register contains the status of interrupts.

Diagram



Fields

Field	Function
31-21 —	Reserved
20 —	Reserved
19 —	Reserved
18 MDIOIS	MDIO Interrupt Status MDIO Interrupt Status This bit indicates an interrupt event after the completion of MDIO operation. To reset this bit, the application has to read this bit/Write 1 to this bit when RCWE bit of MAC_CSR_SW_Ctrl register is set. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - MDIO Interrupt status not active 1b - MDIO Interrupt status active
17 —	Reserved
16 —	Reserved
15 —	Reserved
14	Receive Status Interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
RXSTSIS	This bit indicates the status of received packets. This bit is set when the RWT bit is set in the MAC_Rx_Tx_Status register. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set) in the MAC_Rx_Tx_Status register. 0b - Receive Interrupt status not active 1b - Receive Interrupt status active
13 TXSTSIS	Transmit Status Interrupt This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the MAC_Rx_Tx_Status register: Excessive Collision (EXCOL) Late Collision (LCOL) Excessive Deferral (EXDEF) Loss of Carrier (LCARR) No Carrier (NCARR) Jabber Timeout (TJT) This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set) in the MAC_Rx_Tx_Status register. 0b - Transmit Interrupt status not active 1b - Transmit Interrupt status active
12 TSIS	TSIS Timestamp Interrupt Status If the Timestamp feature is enabled, this bit is set when any of the following conditions is true: The system time value is equal to or exceeds the value specified in the Target Time High and Low registers. There is an overflow in the Seconds register. The Target Time Error occurred, that is, programmed target time already elapsed. In configurations other than EQOS_CORE, when drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_Tx_Timestamp_Status_Nanoseconds and Mac_TxTimestamp_Status_Seconds registers. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set) in the MAC_Timestamp_Status register. 0b - Timestamp Interrupt status not active 1b - Timestamp Interrupt status active
11 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 —	Reserved
9 —	Reserved
8 —	Reserved
7-6 —	Reserved
5 LPIIS	LPIIS LPI Interrupt Status When the Energy Efficient Ethernet feature is enabled, this bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared when the corresponding interrupt source bit of MAC_LPI_Control_Status register is read (or corresponding interrupt source bit of MAC_LPI_Control_Status register is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set). 0b - LPI Interrupt status not active 1b - LPI Interrupt status active
4 PMTIS	PMTIS PMT Interrupt Status This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in MAC_PMT_Control_Status register). This bit is cleared when corresponding interrupt source bit is cleared because of a Read operation to the MAC_PMT_Control_Status register (or corresponding interrupt source bit of MAC_PMT_Control_Status register is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set). This bit is valid only when you select the Enable Power Management option. 0b - PMT Interrupt status not active 1b - PMT Interrupt status active
3 PHYIS	PHY Interrupt This bit is set when rising edge is detected on the phy_intr_i input. This bit is cleared when this register is read (or this bit is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set). 0b - PHY Interrupt not detected 1b - PHY Interrupt detected
2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 —	Reserved
0 —	Reserved

70.5.1.16 Interrupt Enable (MAC_INTERRUPT_ENABLE)

Offset

Register	Offset
MAC_INTERRUPT_ENABLE	B4h

Function

The Interrupt Enable register contains the masks for generating the interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved													MDIOIE	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	RXSTSIE	TXSTSIE	TSIE	Reserved						LPIIE	PMTIE	PHYIE	Reserved	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-19 —	Reserved
18 MDIOIE	MDIO Interrupt Enable When this bit is set, it enables the assertion of the interrupt when MDIOIS field is set in the MAC_Interrupt_Status register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - MDIO Interrupt is disabled 1b - MDIO Interrupt is enabled
17 —	Reserved
16 —	Reserved
15 —	Reserved
14 RXSTSIE	Receive Status Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the MAC_Interrupt_Status register. 0b - Receive Status Interrupt is disabled 1b - Receive Status Interrupt is enabled
13 TXSTSIE	Transmit Status Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the MAC_Interrupt_Status register. 0b - Timestamp Status Interrupt is disabled 1b - Timestamp Status Interrupt is enabled
12 TSIE	Timestamp Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in MAC_Interrupt_Status register. 0b - Timestamp Interrupt is disabled 1b - Timestamp Interrupt is enabled
11-6 —	Reserved
5 LPIIE	LPI Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in MAC_Interrupt_Status register. 0b - LPI Interrupt is disabled 1b - LPI Interrupt is enabled
4 PMTIE	PMT Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in MAC_Interrupt_Status register. 0b - PMT Interrupt is disabled 1b - PMT Interrupt is enabled
3 PHYIE	PHY Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in MAC_Interrupt_Status register. 0b - PHY Interrupt is disabled 1b - PHY Interrupt is enabled
2 —	Reserved
1 —	Reserved
0 —	Reserved

70.5.1.17 Receive Transmit Status (MAC_RX_TX_STATUS)

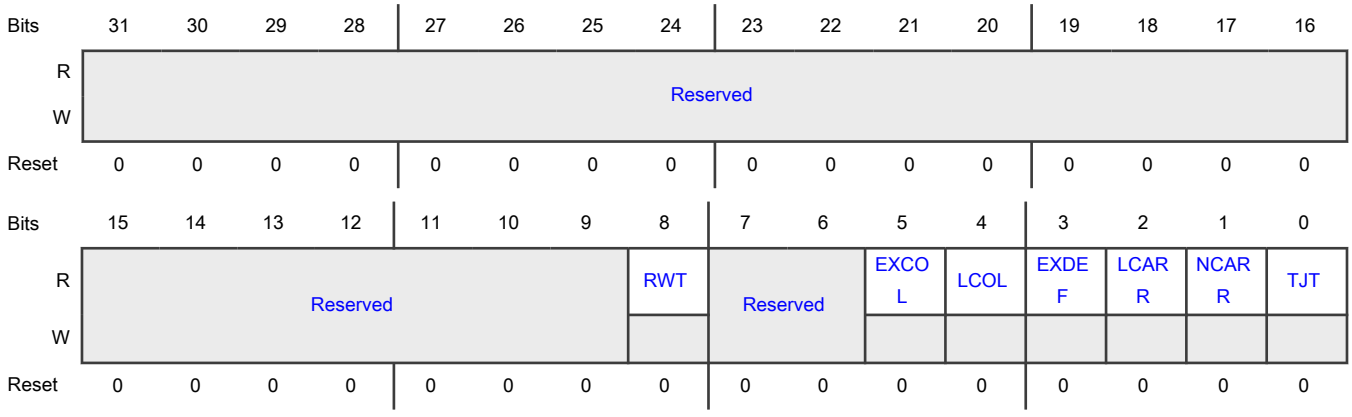
Offset

Register	Offset
MAC_RX_TX_STATUS	B8h

Function

The Receive Transmit Status register contains the Receive and Transmit Error status.

Diagram



Fields

Field	Function
31-9 —	Reserved
8 RWT	Receive Watchdog Timeout This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the MAC_Configuration register. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - No receive watchdog timeout 1b - Receive watchdog timed out
7-6 —	Reserved
5 EXCOL	Excessive Collisions When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after the first collision and the packet transmission is aborted. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - No collision 1b - Excessive collision is sensed
4 LCOL	Late Collision When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode; 512 bytes including Preamble and Carrier Extension in GMII mode). This bit is not valid if the Underflow error occurs. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - No collision 1b - Late collision is sensed
3 EXDEF	Excessive Deferral When the DTXSTS bit is set in the MTL_Operation_Mode register and the DC bit is set in the MAC_Configuration register, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000/2500 Mbps mode or when Jumbo packet is enabled). Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No Excessive deferral 1b - Excessive deferral
2 LCARR	Loss of Carrier When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the loss of carrier occurred during packet transmission, that is, the phy_crs_i signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - Carrier is present 1b - Loss of carrier
1 NCARR	No Carrier When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the carrier signal from the PHY is not present at the end of the preamble transmission. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - Carrier is present 1b - No carrier
0 TJT	Transmit Jabber Timeout This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the MAC_Configuration register. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - No Transmit Jabber Timeout 1b - Transmit Jabber Timeout occurred

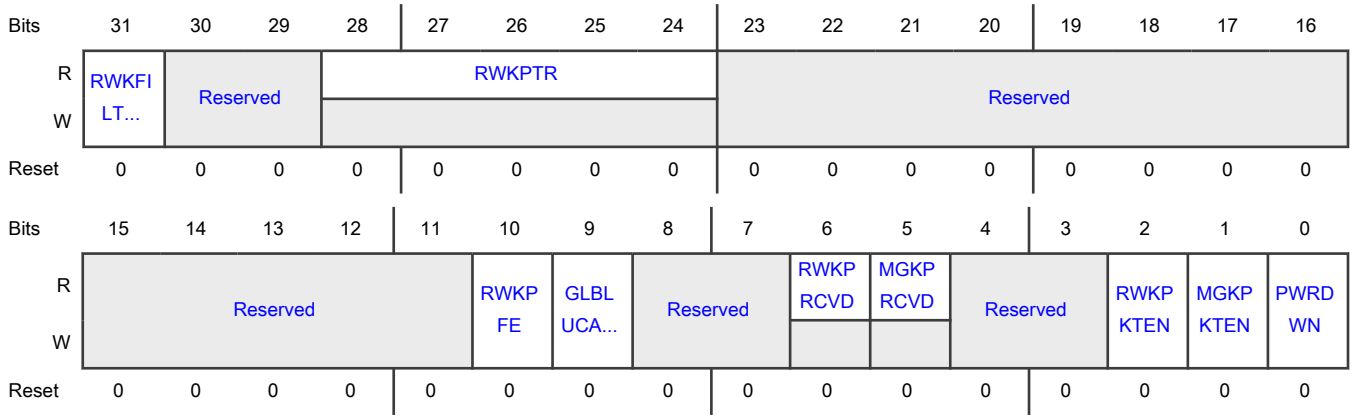
70.5.1.18 PMT Control and Status (MAC_PMT_CONTROL_STATUS)

Offset

Register	Offset
MAC_PMT_CONTROL_STATUS	C0h

Function
The PMT Control and Status Register.

Diagram



Fields

Field	Function
31 RWKFILTRST	Remote Wake-Up Packet Filter Register Pointer Reset When this bit is set, the remote wake-up packet filter register pointer is reset to 3'b000. It is automatically cleared after 1 clock cycle. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Remote Wake-Up Packet Filter Register Pointer is not Reset 1b - Remote Wake-Up Packet Filter Register Pointer is Reset
30-29 —	Reserved
28-24 RWKPTR	Remote Wake-up FIFO Pointer This field gives the current value (0 to 7, 15, or 31 when 4, 8, or 16 Remote Wake-up Packet Filters are selected) of the Remote Wake-up Packet Filter register pointer. When the value of this pointer is equal to maximum for the selected number of Remote Wake-up Packet Filters, the contents of the Remote Wake-up Packet Filter Register are transferred to the clk_rx_i domain when a Write occurs to that register.
23-11 —	Reserved
10 RWKPFE	Remote Wake-up Packet Forwarding Enable When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected Wake-up frame. All frames after that event including the received wake-up frame are forwarded to the application. This bit is then self-cleared on receiving the wake-up packet. The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Note: If Magic Packet Enable and Wake-Up Frame Enable are both set along with the setting of this bit and Magic Packet is received before wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to the application. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Remote Wake-up Packet Forwarding is disabled 1b - Remote Wake-up Packet Forwarding is enabled
9 GLBLUCAST	Global Unicast When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet. 0b - Global unicast is disabled 1b - Global unicast is enabled
8-7 —	Reserved
6 RWKPRCVD	Remote Wake-Up Packet Received When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - Remote wake-up packet is received 1b - Remote wake-up packet is received
5 MGKPRCVD	Magic Packet Received When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - No Magic packet is received 1b - Magic packet is received
4-3 —	Reserved
2 RWKPKTEN	Remote Wake-Up Packet Enable When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet. 0b - Remote wake-up packet is disabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Remote wake-up packet is enabled
1 MGKPKTEN	Magic Packet Enable When this bit is set, a power management event is generated when the MAC receives a magic packet. 0b - Magic Packet is disabled 1b - Magic Packet is enabled
0 PWRDWN	Power Down When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote Wake-Up Packet Enable bit is set high. Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the software cannot clear this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Power down is disabled 1b - Power down is enabled

70.5.1.19 Remote Wakeup Filter (MAC_RWK_PACKET_FILTER)

Offset

Register	Offset
MAC_RWK_PACKET_FILTER	C4h

Function

The Remote Wakeup Filter registers are implemented as 8, 16, or 32 indirect access registers (wkuppktfilter_reg#i) based on whether 4, 8, or 16 Remote Wakeup Filters are selected in the configuration and accessed by application through MAC_RWK_Packet_Filter register. When the Remote Wakeup Filters are to be programmed, the entire set of wkuppktfilter_reg registers must be written. The wkuppktfilter_reg register is programmed by sequentially writing the eight, sixteen, or thirty-two register values in MAC_RWK_Packet_Filter register for wkuppktfilter_reg0, wkuppktfilter_reg1, ..., wkuppktfilter_reg31 respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in the RWKPTR field of the MAC_PMT_Control_Status register.

The Remote Wakeup Filters are arranged in blocks of 4 filters each, and each such block has eight 32-bit wide registers, that is, wkuppktfilter_reg0-7, wkuppktfilter_reg8-15, wkuppktfilter_reg16-23 and wkuppktfilter_reg24-31. The fields of Remote Wakeup Filter are described as follows:

Filter i Byte Mask: The filter i byte mask register defines the bytes of the packet that are examined by filter i (0, 1, 2, 3, .., 15) to determine whether or not a packet is a wake-up packet.

The MSB (31st bit) must be zero.

Bit j[30:0] is the byte mask.

If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter i Offset + j of the incoming packet; otherwise Filter i Offset + j is ignored.

Filter i Command: The 4-bit filter i command controls the filter i operation.

Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet.

Bit 2 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC-16 value. Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2".

Bit 1 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. The details are,

- The And_Previous bit setting is applicable within a set of 4 filters.

- Setting of And_Previous bit of filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of 4 filters has no effect. For example, setting of And_Previous bit of Filter 0 has no effect.

- If And_Previous bit is set for filter to form AND chained filter, the AND chain breaks at the point any filter is not enabled. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting of Filter 2 And_Previous bit has no effect only Filter 1 result ORed with Filter 3 result is considered.

- If filters chained by And_Previous bit setting have complementary programming, then a frame might never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, a remote wakeup frame does not pass the AND chained filter as a remote wakeup frame cannot be of both unicast and multicast address type.

Bit 0 is the enable for filter i. If Bit 0 is not set, filter i is disabled.

Filter i Offset: The filter i offset register defines the offset (within the packet) from which the filter i examines the packets.

This 8-bit pattern-offset is the offset for the filter i first byte to be examined.

The minimum allowed offset is 12, which refers to the 13th byte of the packet.

The offset value 0 refers to the first byte of the packet.

Filter i CRC-16: The filter i CRC-16 register contains the CRC-16 value calculated from the pattern and the byte mask programmed in the Remote Wakeup filter register.

The 16-bit CRC calculation uses the following polynomial: G(x) = x^16 + x^15 + x^2 + 1

Each mask, used in the hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following:

- 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1, the corresponding byte is taken into the CRC-16 calculation.

- 8-bit Offset Pointer: Specifies the byte to start the CRC-16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC-16 calculations.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WKUPFRMFTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WKUPFRMFTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 WKUPFRMFTR	RWK Packet Filter This field contains the various controls of RWK Packet filter.

70.5.1.20 LPI Control and Status (MAC_LPI_CONTROL_STATUS)

Offset

Register	Offset
MAC_LPI_CONTROL_STATUS	D0h

Function

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status.

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved										LPITCSE	LPIATE	LPITXA	Reserved	PLS	LPIEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						RLPIST	TLPIST	Reserved				RLPIEX	RLPIEN	TLPIEX	TLPIEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-22 —	Reserved
21 LPITCSE	LPI Tx Clock Stop Enable When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode. If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed. 0b - LPI Tx Clock Stop is disabled 1b - LPI Tx Clock Stop is enabled
20 LPIATE	LPI Timer Enable This bit controls the automatic entry of the MAC Transmitter into and exit out of the LPI state. When LPIATE, LPITXA and LPIEN bits are set, the MAC Transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the MAC_LPI_Entry_Timer register. After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the Transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again. When LPIATE is 0, the LPI Auto timer is disabled and MAC Transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions. 0b - LPI Timer is disabled 1b - LPI Timer is enabled
19 LPITXA	LPI Tx Automate This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side. This bit is not functional in the EQOS-CORE configurations in which the Tx clock gating is done during the LPI mode. If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the IP) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of MTL_TxQ0_Operation_Mode register, when the MAC is in the LPI mode, it exits the LPI mode. When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode. 0b - LPI Tx Automate is disabled 1b - LPI Tx Automate is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 —	Reserved
17 PLS	PHY Link Status This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER. When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down. 0b - link is down 1b - link is okay (UP)
16 LPIEN	LPI Enable When this bit is set, it instructs the MAC Transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission. This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission. 0b - LPI state is disabled 1b - LPI state is enabled
15-10 —	Reserved
9 RLPIST	Receive LPI State When this bit is set, it indicates that the MAC is receiving the LPI pattern on the GMII or MII interface. 0b - Receive LPI state not detected 1b - Receive LPI state detected
8 TLPIST	Transmit LPI State When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface. 0b - Transmit LPI state not detected 1b - Transmit LPI state detected
7-4 —	Reserved
3 RLPIEX	Receive LPI Exit When this bit is set, it indicates that the MAC Receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Note: This bit might not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock. 0b - Receive LPI exit not detected 1b - Receive LPI exit detected
2 RLPIEN	Receive LPI Entry When this bit is set, it indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Note: This bit might not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock. 0b - Receive LPI entry not detected 1b - Receive LPI entry detected
1 TLPIEX	Transmit LPI Exit When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). 0b - Transmit LPI exit not detected 1b - Transmit LPI exit detected
0 TLPIEN	Transmit LPI Entry When this bit is set, it indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). 0b - Transmit LPI entry not detected 1b - Transmit LPI entry detected

70.5.1.21 LPI Timers Control (MAC_LPI_TIMERS_CONTROL)

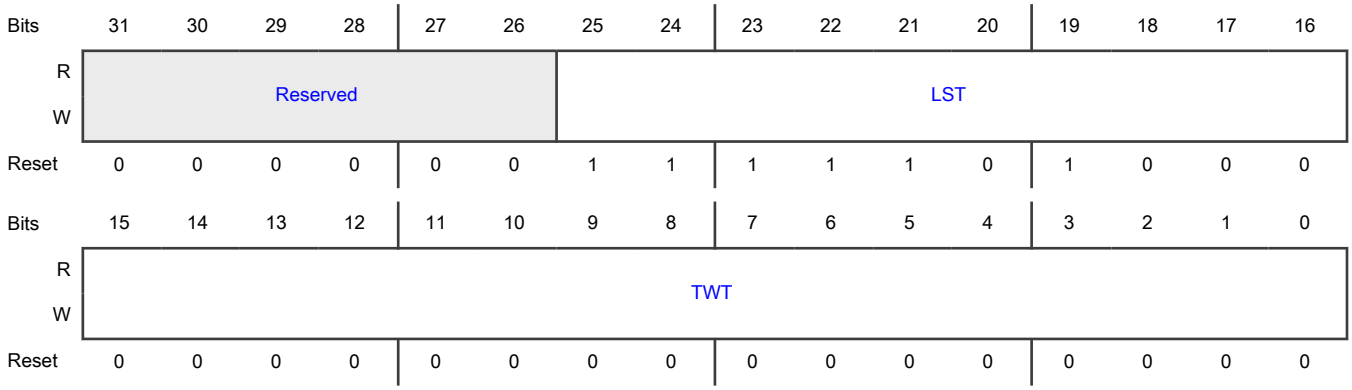
Offset

Register	Offset
MAC_LPI_TIMERS_CONTROL	D4h

Function

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 LST	LPI LS Timer This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.
15-0 TWT	LPI TW Timer This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

70.5.1.22 Tx LPI Entry Timer Control (MAC_LPI_ENTRY_TIMER)

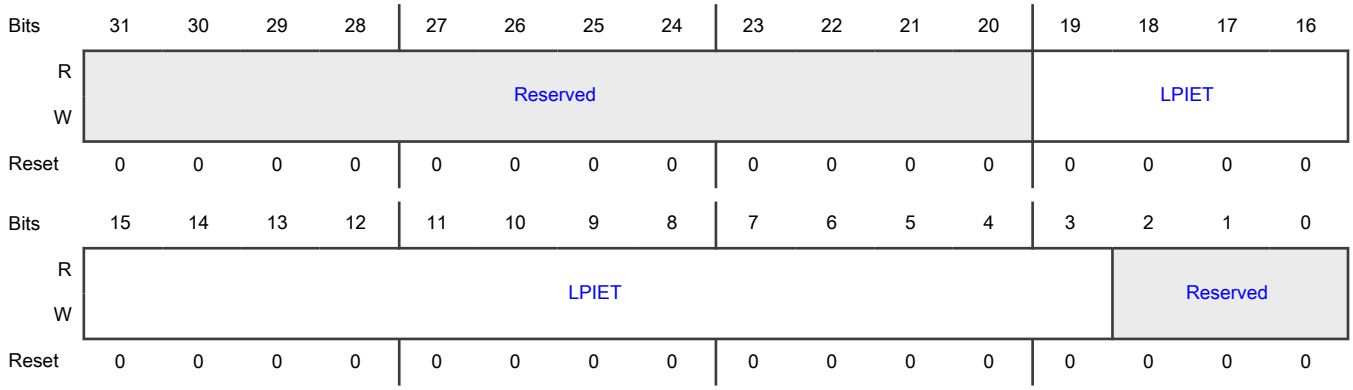
Offset

Register	Offset
MAC_LPI_ENTRY_TIMER	D8h

Function

This register controls the Tx LPI entry timer. This counter is enabled only when bit[20](LPITE) bit of MAC_LPI_Control_Status is set to 1.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-3 LPIET	LPI Entry Timer This field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1. Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.
2-0 —	Reserved

70.5.1.23 One-microsecond Reference Timer (MAC_ONEUS_TIC_COUNTER)

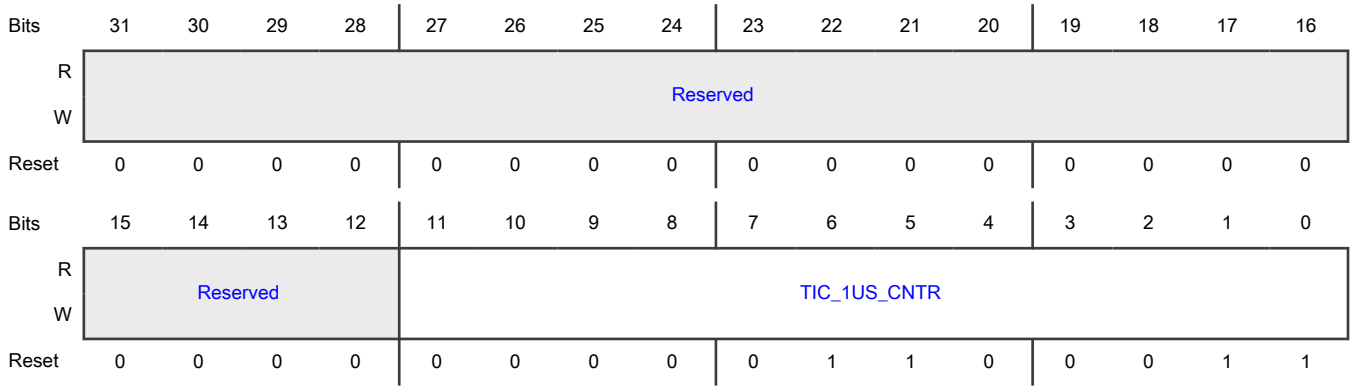
Offset

Register	Offset
MAC_ONEUS_TIC_COUNTER	DCh

Function

This register controls the generation of the Reference time (1 microsecond tic) for all the LPI timers. This timer has to be programmed by the software initially.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-0 TIC_1US_CNTR	1US TIC Counter The application must program this counter so that the number of clock cycles of CSR clock is 1us. (Subtract 1 from the value before programming). For example if the CSR clock is 100 MHz then this field needs to be programmed to value 100 - 1 = 99 (which is 0x63). This is required to generate the 1US events that are used to update some of the EEE related counters.

70.5.1.24 MAC Version (MAC_VERSION)

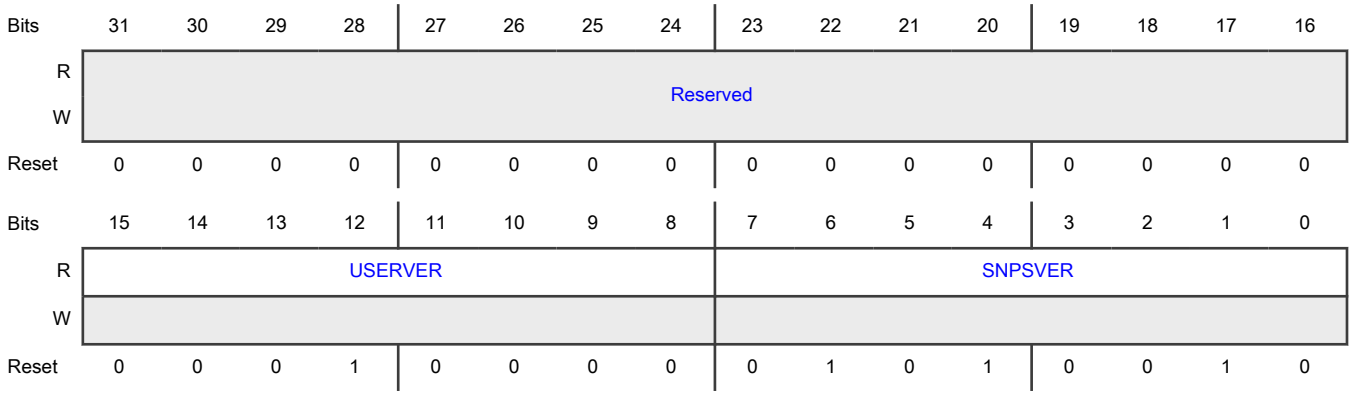
Offset

Register	Offset
MAC_VERSION	110h

Function

The version register identifies the version of the enet. This register contains two bytes: one that Synopsys uses to identify the IP release number, and the other that you set while configuring the IP.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 USERVER	User-defined Version
7-0 SNPSVER	Synopsys-defined Version

70.5.1.25 MAC Debug (MAC_DEBUG)

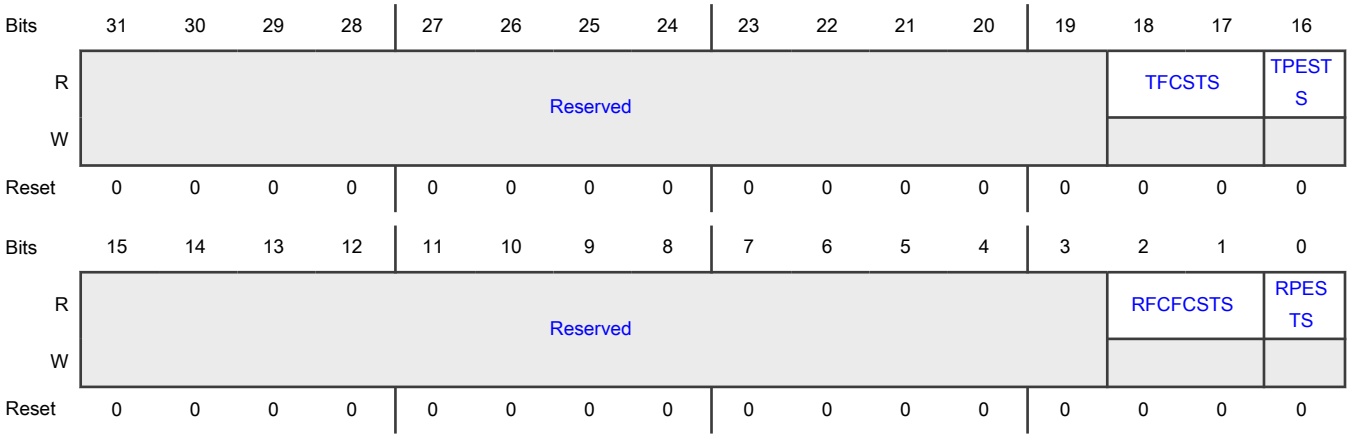
Offset

Register	Offset
MAC_DEBUG	114h

Function

The Debug register provides the debug status of various MAC blocks.

Diagram



Fields

Field	Function
31-19 —	Reserved
18-17 TFCSTS	MAC Transmit Packet Controller Status This field indicates the state of the MAC Transmit Packet Controller module. 00b - Idle state 01b - Waiting for one of the following: Status of the previous packet OR IPG or back off period to be over 10b - Generating and transmitting a Pause control packet (in full-duplex mode) 11b - Transferring input packet for transmission
16 TPESTS	MAC GMII or MII Transmit Protocol Engine Status When this bit is set, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state. 0b - MAC GMII or MII Transmit Protocol Engine Status not detected 1b - MAC GMII or MII Transmit Protocol Engine Status detected
15-3 —	Reserved
2-1 RFCFCSTS	MAC Receive Packet Controller FIFO Status When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.
0 RPESTS	MAC GMII or MII Receive Protocol Engine Status When this bit is set, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data, and it is not in the Idle state.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - MAC GMII or MII Receive Protocol Engine Status not detected 1b - MAC GMII or MII Receive Protocol Engine Status detected

70.5.1.26 Hardware Features 0 (MAC_HW_FEATURE0)

Offset

Register	Offset
MAC_HW_FEATURE0	11Ch

Function

This register indicates the presence of the first set of the optional features or functions of the enet. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	ACTPHYSEL			SAVLANI...	TSSTSSEL		MACADR6...	MACADR3...	ADDMACADRSEL					Reserved	RXCOESEL
W																
Reset	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	TXCOESEL	EEESEL	TSSEL	Reserved		ARPOFFS...	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
W																
Reset	0	1	1	1	0	0	0	0	1	1	1	0	0	1	0	1

Fields

Field	Function
31 —	Reserved
30-28 ACTPHYSEL	Active PHY Selected When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion. 000b - GMII or MII

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001b - RGMII 010b - SGMII 011b - TBI 100b - RMII 101b - RTBI 110b - SMII 111b - RevMII
27 SAVLANINS	Source Address or VLAN Insertion Enable This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected 0b - Source Address or VLAN Insertion Enable option is not selected 1b - Source Address or VLAN Insertion Enable option is selected
26-25 TSSTSSEL	Timestamp System Time Source This bit indicates the source of the Timestamp system time: This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected 00b - Internal 01b - External 10b - Both 11b - Reserved
24 MACADR64SEL	MAC Addresses 64-127 Selected This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected 0b - MAC Addresses 64-127 Select option is not selected 1b - MAC Addresses 64-127 Select option is selected
23 MACADR32SEL	MAC Addresses 32-63 Selected This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected 0b - MAC Addresses 32-63 Select option is not selected 1b - MAC Addresses 32-63 Select option is selected
22-18 ADDMACADRSEL	MAC Addresses 1-31 Selected This bit is set to 1 when the non-zero value is selected for Enable Additional 1-31 MAC Address Registers option
17 —	Reserved
16	Receive Checksum Offload Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
RXCOESEL	This bit is set to 1 when the Enable Receive TCP/IP Checksum Check option is selected 0b - Receive Checksum Offload Enable option is not selected 1b - Receive Checksum Offload Enable option is selected
15 —	Reserved
14 TXCOESEL	Transmit Checksum Offload Enabled This bit is set to 1 when the Enable Transmit TCP/IP Checksum Insertion option is selected 0b - Transmit Checksum Offload Enable option is not selected 1b - Transmit Checksum Offload Enable option is selected
13 EEESEL	Energy Efficient Ethernet Enabled This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected 0b - Energy Efficient Ethernet Enable option is not selected 1b - Energy Efficient Ethernet Enable option is selected
12 TSSEL	IEEE 1588-2008 Timestamp Enabled This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected 0b - IEEE 1588-2008 Timestamp Enable option is not selected 1b - IEEE 1588-2008 Timestamp Enable option is selected
11-10 —	Reserved
9 ARPOFFSEL	ARP Offload Enabled This bit is set to 1 when the Enable IPv4 ARP Offload option is selected 0b - ARP Offload Enable option is not selected 1b - ARP Offload Enable option is selected
8 MMCSEL	RMON Module Enable This bit is set to 1 when the Enable MAC Management Counters (MMC) option is selected 0b - RMON Module Enable option is not selected 1b - RMON Module Enable option is selected
7 MGKSEL	PMT Magic Packet Enable This bit is set to 1 when the Enable Magic Packet Detection option is selected 0b - PMT Magic Packet Enable option is not selected 1b - PMT Magic Packet Enable option is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 RWKSEL	PMT Remote Wake-up Packet Enable This bit is set to 1 when the Enable Remote Wake-Up Packet Detection option is selected 0b - PMT Remote Wake-up Packet Enable option is not selected 1b - PMT Remote Wake-up Packet Enable option is selected
5 SMASEL	SMA (MDIO) Interface This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected 0b - SMA (MDIO) Interface not selected 1b - SMA (MDIO) Interface selected
4 VLHASH	VLAN Hash Filter Selected This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected 0b - VLAN Hash Filter not selected 1b - VLAN Hash Filter selected
3 PCSSEL	PCS Registers (TBI, SGMII, or RTBI PHY interface) This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected 0b - No PCS Registers (TBI, SGMII, or RTBI PHY interface) 1b - PCS Registers (TBI, SGMII, or RTBI PHY interface)
2 HDSEL	Half-duplex Support This bit is set to 1 when the half-duplex mode is selected 0b - No Half-duplex support 1b - Half-duplex support
1 GMIISEL	1000 Mbps Support This bit is set to 1 when 1000 Mbps is selected as the Mode of Operation 0b - No 1000 Mbps support 1b - 1000 Mbps support
0 MIISEL	10 or 100 Mbps Support This bit is set to 1 when 10/100 Mbps is selected as the Mode of Operation 0b - No 10 or 100 Mbps support 1b - 10 or 100 Mbps support

70.5.1.27 Hardware Features 1 (MAC_HW_FEATURE1)

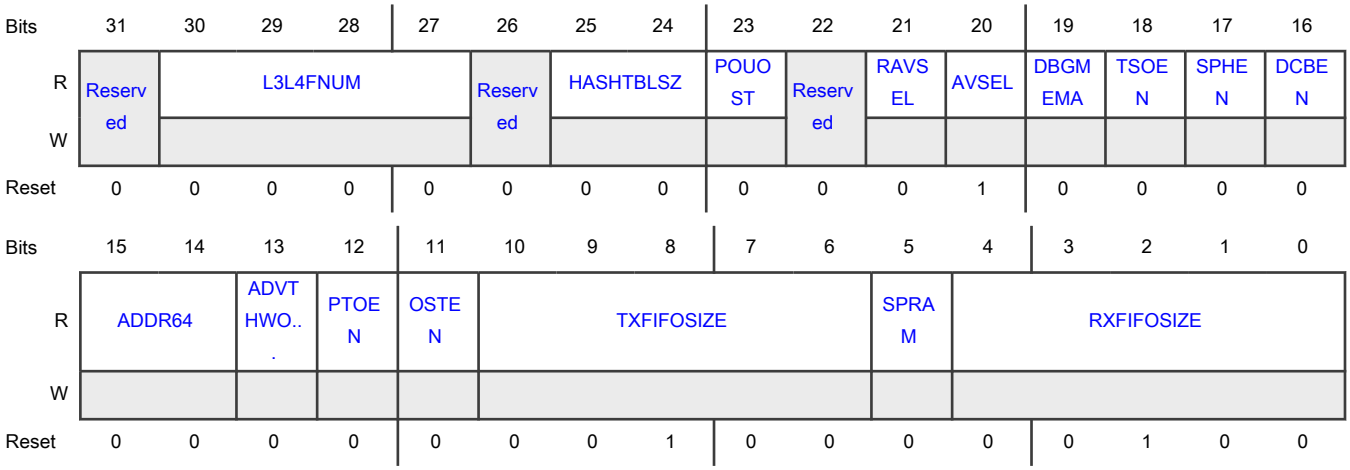
Offset

Register	Offset
MAC_HW_FEATURE1	120h

Function

This register indicates the presence of the second set of the optional features or functions of the enet. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Diagram



Fields

Field	Function
31 —	Reserved
30-27 L3L4FNUM	Total number of L3 or L4 Filters This field indicates the total number of L3 or L4 filters: 0000b - No L3 or L4 Filter 0001b - 1 L3 or L4 Filter 0010b - 2 L3 or L4 Filters 0011b - 3 L3 or L4 Filters 0100b - 4 L3 or L4 Filters 0101b - 5 L3 or L4 Filters 0110b - 6 L3 or L4 Filters

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0111b - 7 L3 or L4 Filters 1000b - 8 L3 or L4 Filters
26 —	Reserved
25-24 HASHTBLSZ	Hash Table Size This field indicates the size of the hash table: 00b - No hash table 01b - 64 10b - 128 11b - 256
23 POUOST	One Step for PTP over UDP/IP Feature Enable This bit is set to 1 when the Enable One step timestamp for PTP over UDP/IP feature is selected. 0b - One Step for PTP over UDP/IP Feature is not selected 1b - One Step for PTP over UDP/IP Feature is selected
22 —	Reserved
21 RAVSEL	Rx Side Only AV Feature Enable This bit is set to 1 when the Enable Audio Video Bridging option on Rx Side Only is selected. 0b - Rx Side Only AV Feature is not selected 1b - Rx Side Only AV Feature is selected
20 AVSEL	AV Feature Enable This bit is set to 1 when the Enable Audio Video Bridging option is selected. 0b - AV Feature is not selected 1b - AV Feature is selected
19 DBGMEMA	DMA Debug Registers Enable This bit is set to 1 when the Debug Mode Enable option is selected 0b - DMA Debug Registers option is not selected 1b - DMA Debug Registers option is selected
18 TSOEN	TCP Segmentation Offload Enable This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - TCP Segmentation Offload Feature is not selected 1b - TCP Segmentation Offload Feature is selected
17 SPHEN	Split Header Feature Enable This bit is set to 1 when the Enable Split Header Structure option is selected 0b - Split Header Feature is not selected 1b - Split Header Feature is selected
16 DCBEN	DCB Feature Enable This bit is set to 1 when the Enable Data Center Bridging option is selected 0b - DCB Feature is not selected 1b - DCB Feature is selected
15-14 ADDR64	Address Width. Address Width. This field indicates the configured address width: 00b - 32 01b - 40 10b - 48 11b - Reserved
13 ADVTHWORD	IEEE 1588 High Word Register Enable This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected 0b - IEEE 1588 High Word Register option is not selected 1b - IEEE 1588 High Word Register option is selected
12 PTOEN	PTP Offload Enable This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected. 0b - PTP Offload feature is not selected 1b - PTP Offload feature is selected
11 OSTEN	One-Step Timestamping Enable This bit is set to 1 when the Enable One-Step Timestamp Feature is selected. 0b - One-Step Timestamping feature is not selected 1b - One-Step Timestamping feature is selected
10-6 TXFIFOSIZE	MTL Transmit FIFO Size This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0000b - 128 bytes 0_0001b - 256 bytes 0_0010b - 512 bytes 0_0011b - 1024 bytes 0_0100b - 2048 bytes 0_0101b - 4096 bytes 0_0110b - 8192 bytes 0_0111b - 16384 bytes 0_1000b - 32 KB 0_1001b - 64 KB 0_1010b - 128 KB 0_1011b - Reserved
5 SPRAM	Single Port RAM Enable This bit is set to 1 when the Use single port RAM Feature is selected. 0b - Single Port RAM feature is not selected 1b - Single Port RAM feature is selected
4-0 RXFIFOSIZE	MTL Receive FIFO Size This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7: 0_0000b - 128 bytes 0_0001b - 256 bytes 0_0010b - 512 bytes 0_0011b - 1024 bytes 0_0100b - 2048 bytes 0_0101b - 4096 bytes 0_0110b - 8192 bytes 0_0111b - 16384 bytes 0_1000b - 32 KB 0_1001b - 64 KB 0_1010b - 128 KB 0_1011b - 256 KB 0_1100b - Reserved

70.5.1.28 Hardware Features 2 (MAC_HW_FEATURE2)

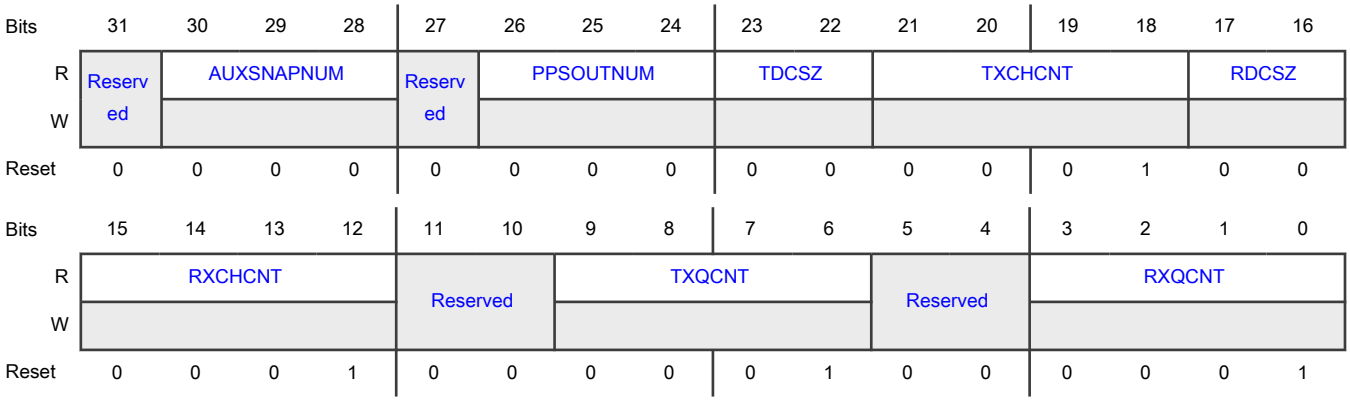
Offset

Register	Offset
MAC_HW_FEATURE2	124h

Function

This register indicates the presence of the third set of the optional features or functions of the enet. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Diagram



Fields

Field	Function
31 —	Reserved
30-28 AUXSNAPNUM	Number of Auxiliary Snapshot Inputs This field indicates the number of auxiliary snapshot inputs: 000b - No auxiliary input 001b - 1 auxiliary input 010b - 2 auxiliary input 011b - 3 auxiliary input 100b - 4 auxiliary input 101b - Reserved
27 —	Reserved
26-24	Number of PPS Outputs

Table continues on the next page...

Table continued from the previous page...

Field	Function
PPSOUTNUM	This field indicates the number of PPS outputs: 000b - No PPS output 001b - 1 PPS output 010b - 2 PPS output 011b - 3 PPS output 100b - 4 PPS output 101b - Reserved
23-22 TDCSZ	Tx DMA Descriptor Cache Size in terms of 16 bytes descriptors: 00 - Cache Not configured 01 - 4 16 bytes descriptor 10 - 8 16 bytes descriptor 11 - 16 16 bytes descriptor
21-18 TXCHCNT	Number of DMA Transmit Channels This field indicates the number of DMA Transmit channels: 0000b - 1 MTL Tx Channel 0001b - 2 MTL Tx Channels 0010b - 3 MTL Tx Channels 0011b - 4 MTL Tx Channels 0100b - 5 MTL Tx Channels 0101b - 6 MTL Tx Channels 0110b - 7 MTL Tx Channels 0111b - 8 MTL Tx Channels
17-16 RDCSZ	Rx DMA Descriptor Cache Size in terms of 16 bytes descriptors: 00 - Cache Not configured 01 - 4 16 bytes descriptor 10 - 8 16 bytes descriptor 11 - 16 16 bytes descriptor
15-12 RXCHCNT	Number of DMA Receive Channels This field indicates the number of DMA Receive channels: 0000b - 1 MTL Rx Channel 0001b - 2 MTL Rx Channels 0010b - 3 MTL Rx Channels

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0011b - 4 MTL Rx Channels 0100b - 5 MTL Rx Channels 0101b - 6 MTL Rx Channels 0110b - 7 MTL Rx Channels 0111b - 8 MTL Rx Channels
11-10 —	Reserved
9-6 TXQCNT	Number of MTL Transmit Queues This field indicates the number of MTL Transmit queues: 0000b - 1 MTL Tx Queue 0001b - 2 MTL Tx Queues 0010b - 3 MTL Tx Queues 0011b - 4 MTL Tx Queues 0100b - 5 MTL Tx Queues 0101b - 6 MTL Tx Queues 0110b - 7 MTL Tx Queues 0111b - 8 MTL Tx Queues
5-4 —	Reserved
3-0 RXQCNT	Number of MTL Receive Queues This field indicates the number of MTL Receive queues: 0000b - 1 MTL Rx Queue 0001b - 2 MTL Rx Queues 0010b - 3 MTL Rx Queues 0011b - 4 MTL Rx Queues 0100b - 5 MTL Rx Queues 0101b - 6 MTL Rx Queues 0110b - 7 MTL Rx Queues 0111b - 8 MTL Rx Queues

70.5.1.29 Hardware Features 3 (MAC_HW_FEATURE3)

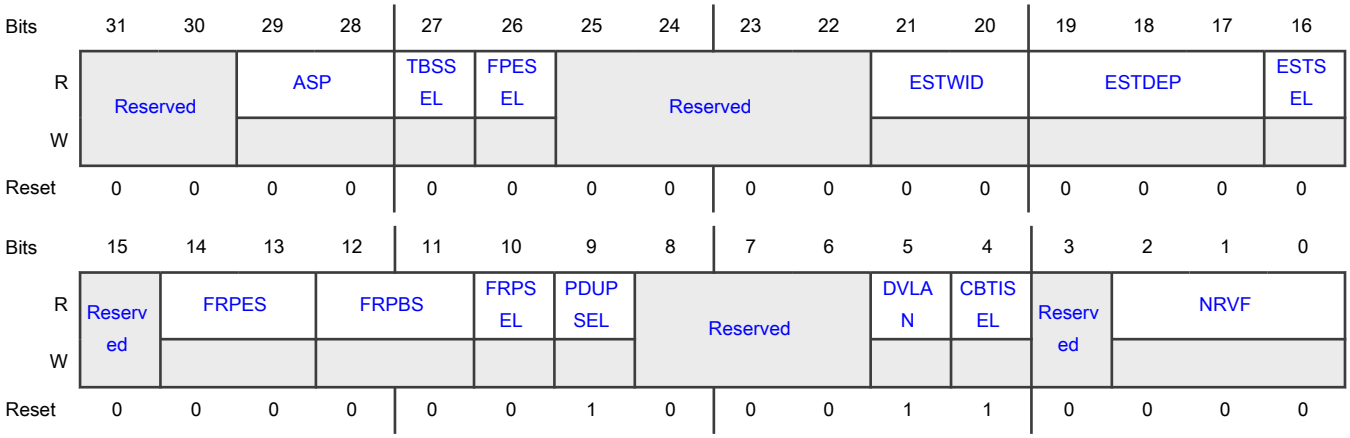
Offset

Register	Offset
MAC_HW_FEATURE3	128h

Function

This register indicates the presence of the fourth set the optional features or functions of the enet. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-28 ASP	Automotive Safety Package Following are the encoding for the different Safety features 00b - No Safety features selected 01b - Only "ECC protection for external memory" feature is selected 10b - All the Automotive Safety features are selected without the "Parity Port Enable for external interface" feature 11b - All the Automotive Safety features are selected with the "Parity Port Enable for external interface" feature
27 TBSSEL	Time Based Scheduling Enable This bit is set to 1 when the Time Based Scheduling feature is selected.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Time Based Scheduling Enable feature is not selected 1b - Time Based Scheduling Enable feature is selected
26 FPESEL	Frame Preemption Enable This bit is set to 1 when the Enable Frame preemption feature is selected. 0b - Frame Preemption Enable feature is not selected 1b - Frame Preemption Enable feature is selected
25-22 —	Reserved
21-20 ESTWID	Width of the Time Interval field in the Gate Control List This field indicates the width of the Configured Time Interval Field 00b - Width not configured 01b - 16 10b - 20 11b - 24
19-17 ESTDEP	Depth of the Gate Control List This field indicates the depth of Gate Control list expressed as Log2(DWC_EQOS_EST_DEP)-5 000b - No Depth configured 001b - 64 010b - 128 011b - 256 100b - 512 101b - 1024 110b - Reserved
16 ESTSEL	Enhancements to Scheduled Traffic Enable This bit is set to 1 when the Enable Enhancements to Scheduling Traffic feature is selected. 0b - Enable Enhancements to Scheduling Traffic feature is not selected 1b - Enable Enhancements to Scheduling Traffic feature is selected
15 —	Reserved
14-13 FRPES	Flexible Receive Parser Table Entries size This field indicates the Max Number of Parser Entries supported by Flexible Receive Parser.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - 64 Entries 01b - 128 Entries 10b - 256 Entries 11b - Reserved
12-11 FRPBS	Flexible Receive Parser Buffer size This field indicates the supported Max Number of bytes of the packet data to be Parsed by Flexible Receive Parser. 00b - 64 Bytes 01b - 128 Bytes 10b - 256 Bytes 11b - Reserved
10 FRPSEL	Flexible Receive Parser Selected This bit is set to 1 when the Enable Flexible Programmable Receive Parser option is selected. 0b - Flexible Receive Parser feature is not selected 1b - Flexible Receive Parser feature is selected
9 PDUPSEL	Broadcast/Multicast Packet Duplication This bit is set to 1 when the Broadcast/Multicast Packet Duplication feature is selected. 0b - Broadcast/Multicast Packet Duplication feature is not selected 1b - Broadcast/Multicast Packet Duplication feature is selected
8-6 —	Reserved
5 DVLAN	Double VLAN Tag Processing Selected This bit is set to 1 when the Enable Double VLAN Processing Feature is selected. 0b - Double VLAN option is not selected 1b - Double VLAN option is selected
4 CBTISEL	Queue/Channel based VLAN tag insertion on Tx Enable This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx Feature is selected. 0b - Enable Queue/Channel based VLAN tag insertion on Tx feature is not selected 1b - Enable Queue/Channel based VLAN tag insertion on Tx feature is selected
3 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0 NRVF	Number of Extended VLAN Tag Filters Enabled This field indicates the Number of Extended VLAN Tag Filters selected: 000b - No Extended Rx VLAN Filters 001b - 4 Extended Rx VLAN Filters 010b - 8 Extended Rx VLAN Filters 011b - 16 Extended Rx VLAN Filters 100b - 24 Extended Rx VLAN Filters 101b - 32 Extended Rx VLAN Filters 110b - Reserved

70.5.1.30 MDIO Address (MAC_MDIO_ADDRESS)

Offset

Register	Offset
MAC_MDIO_ADDRESS	200h

Function

Controls the management cycles to external PHY through a management interface.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				PSE	BTB	PA						RDA			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	NTC			CR				Reserved			SKAP	GOC_1	GOC_0	C45E	GB
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
27 PSE	Preamble Suppression Enable Preamble Suppression Enable When this bit is set, the SMA suppresses the 32-bit preamble and transmits MDIO frames with only 1 preamble bit. When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications. 0b - Preamble Suppression disabled 1b - Preamble Suppression enabled
26 BTB	Back to Back transactions When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame. When this bit is reset, then the read/write command completion (GB is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame. This bit must not be set when NTC=0. 0b - Back to Back transactions disabled 1b - Back to Back transactions enabled
25-21 PA	Physical Layer Address This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing. This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.
20-16 RDA	Register/Device Address These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.
15 —	Reserved
14-12 NTC	NTC Number of Trailing Clocks This field controls the number of trailing clock cycles generated on gmii_mdc_o (MDC) after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3'h3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 CR	CR CSR Clock Range The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design: 0000: CSR clock = 60-100 MHz; MDC clock = CSR clock/42 0001: CSR clock = 100-150 MHz; MDC clock = CSR clock/62 0010: CSR clock = 20-35 MHz; MDC clock = CSR clock/16 0011: CSR clock = 35-60 MHz; MDC clock = CSR clock/26 0100: CSR clock = 150-250 MHz; MDC clock = CSR clock/102 0101: CSR clock = 250-300 MHz; MDC clock = CSR clock/124 0110: CSR clock = 300-500 MHz; MDC clock = CSR clock/204 0111: CSR clock = 500-800 MHz; MDC clock = CSR clock/324 The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits as 1010, the resultant MDC clock is of 12.5 MHz which is beyond the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks: 1000: CSR clock/4 1001: CSR clock/6 1010: CSR clock/8 1011: CSR clock/10 1100: CSR clock/12 1101: CSR clock/14 1110: CSR clock/16 1111: CSR clock/18 These bits are not used for accessing RevMII. These bits are read-only if the RevMII interface is selected as single PHY interface.
7-5 —	Reserved
4 SKAP	Skip Address Packet Skip Address Packet When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Skip Address Packet is disabled 1b - Skip Address Packet is enabled
3 GOC_1	GMII Operation Command 1 This bit is higher bit of the operation command to the PHY or RevMII, GOC_1 and GOC_O is encoded as follows: 00: Reserved 01: Write 10: Post Read Increment Address for Clause 45 PHY 11: Read When Clause 22 PHY or RevMII is enabled, only Write and Read commands are valid. 0b - GMII Operation Command 1 is disabled 1b - GMII Operation Command 1 is enabled
2 GOC_0	GMII Operation Command 0 GMII Operation Command 0 This is the lower bit of the operation command to the PHY or RevMII. When in SMA mode (MDIO master) this bit along with GOC_1 determines the operation to be performed to the PHY. When only RevMII is selected in configuration this bit is read-only and tied to 1. 0b - GMII Operation Command 0 is disabled 1b - GMII Operation Command 0 is enabled
1 C45E	Clause 45 PHY Enable Clause 45 PHY Enable When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO. 0b - Clause 45 PHY is disabled 1b - Clause 45 PHY is enabled
0 GB	GMII Busy GMII Busy The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIO slave. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in MAC_MDIO_Address and MAC_MDIO_Data registers as long as this bit is set. For write transfers, the application must first write 16-bit data in the GDI field (and also RA field when C45E is set) in MAC_MDIO_Data register before

Table continues on the next page...

Table continued from the previous page...

Field	Function
	setting this bit. When C45E is set, it should also write into the RA field of MAC_MDIO_Data register before initiating a read transfer. When a read transfer is completed (GB=0), the data read from the PHY register is valid in the GD field of the MAC_MDIO_Data register. Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - GMII Busy is disabled 1b - GMII Busy is enabled

70.5.1.31 MAC MDIO Data (MAC_MDIO_DATA)

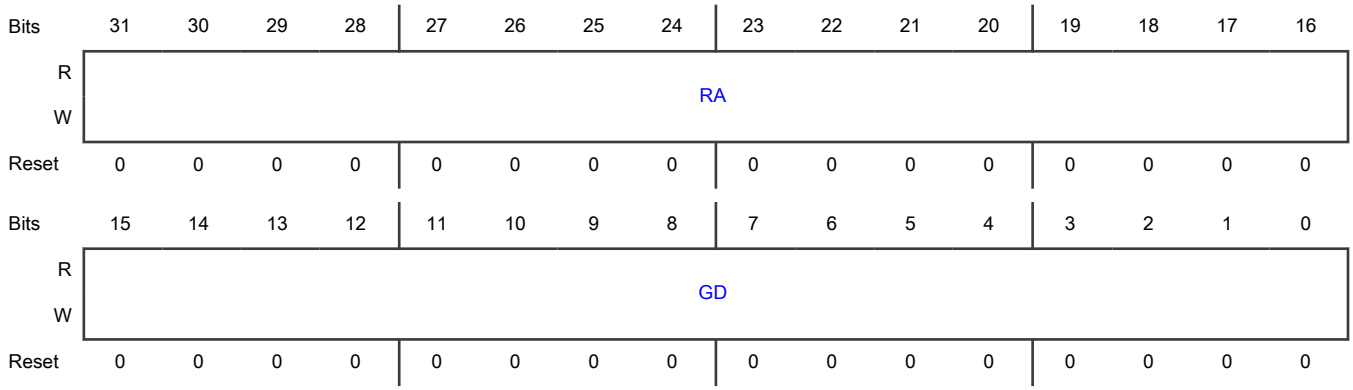
Offset

Register	Offset
MAC_MDIO_DATA	204h

Function

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in MAC_MDIO_Address. This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

Diagram



Fields

Field	Function
31-16	Register Address

Table continues on the next page...

Table continued from the previous page...

Field	Function
RA	This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.
15-0 GD	GMII Data This field contains the 16-bit data value read from the PHY or RevMII after a Management Read operation or the 16-bit data value to be written to the PHY or RevMII before a Management Write operation.

70.5.1.32 CSR Software Control (MAC_CSR_SW_CTRL)

Offset

Register	Offset
MAC_CSR_SW_CTRL	230h

Function

This register contains SW programmable controls for changing the CSR access response and status bits clearing.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							Reserved	Reserved							RCWE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-9 —	Reserved
8 —	Reserved
7-1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 RCWE	Register Clear on Write 1 Enable When this bit is set, the access mode of some register fields changes to Clear on Write 1, the application needs to set that respective bit to 1 to clear it. When this bit is reset, the access mode of these register fields remain as Clear on Read. 0b - Register Clear on Write 1 is disabled 1b - Register Clear on Write 1 is enabled

70.5.1.33 MAC Address0 High (MAC_ADDRESS0_HIGH)

Offset

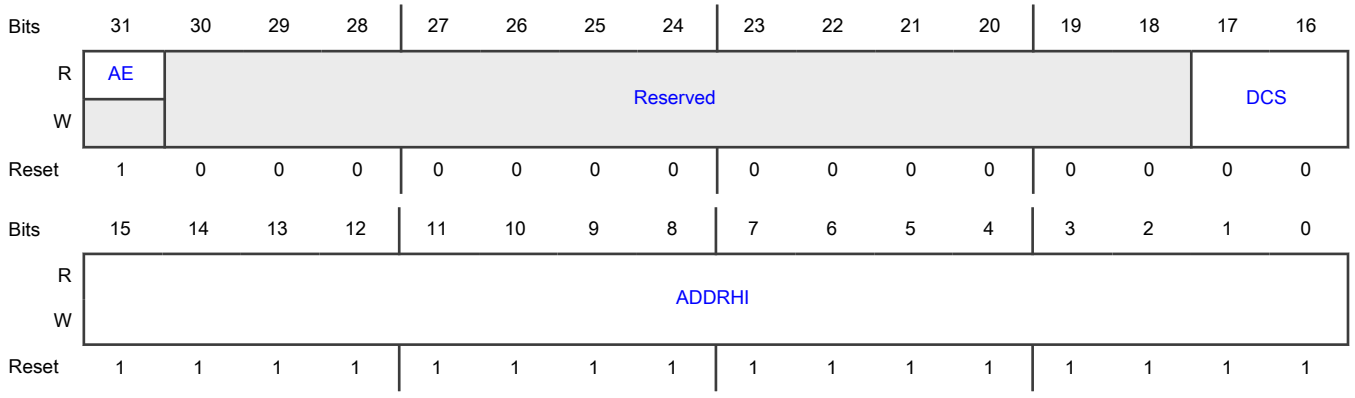
Register	Offset
MAC_ADDRESS0_HIGH	300h

Function

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211.

If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

Diagram



Fields

Field	Function
31 AE	Address Enable This bit is always set to 1. 0b - INVALID : This bit must be always set to 1 1b - This bit is always set to 1
30-18 —	Reserved
17-16 DCS	DMA Channel Select If the PDC bit of MAC_Ext_Configuration register is not set: This field contains the binary representation of the DMA Channel number to which an Rx packet whose DA matches the MAC Address0 content is routed. If the PDC bit of MAC_Ext_Configuration register is set: This field contains the one-hot representation of one or more DMA Channel numbers to which an Rx packet whose DA matches the MAC Address0 content is routed.
15-0 ADDRHI	MAC Address0[47:32] This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

70.5.1.34 MAC Address0 Low (MAC_ADDRESS0_LOW)

Offset

Register	Offset
MAC_ADDRESS0_LOW	304h

Function

The MAC Address0 Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

Diagram

Fields

Field	Function
31-0 ADDRLO	MAC Address0[31:0] This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

70.5.1.35 Indirect Access Control (INDIR_ACCESS_CTRL)

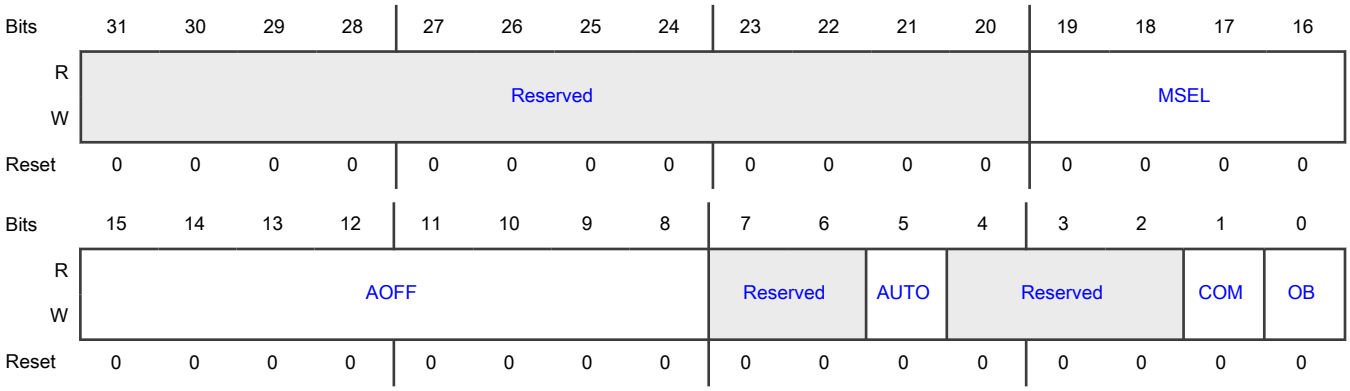
Offset

Register	Offset
INDIR_ACCESS_CTRL	A70h

Function

This register provides the Indirect Access control and status for MAC registers.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 MSEL	Mode Select This field is used in indirect access of MAC_<MSEL>_<AOFF>. This field must be set along with initiation of read/write to MAC_<MSEL>_<AOFF> and should not be changed until the OB is reset. Values: 0000:Typ_RXQ_ (Type based RXQ mapping) 0001-1111: Reserved for future use.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-8 AOFF	Address Offset Address Offset This field is used in indirect access of MAC_<MSEL>_<AOFF>. This field must be set along with initiation of read/write to MAC_<MSEL>_<AOFF> and should not be changed until the OB is reset. Values: 00000000: IndReg0(Indirect register 0) 00000001: IndReg1(Indirect register 1) ... 00000111: IndReg7(Indirect register 7) 00001000-11111111: Reserved for future use.
7-6 —	Reserved
5 AUTO	Auto increment 1: AOFF is incremented by 1. Software should ensure not to cause a wrap condition. Byte wise read/write is not supported when auto increment is enabled. 0: AOFF is not incremented automatically. Software should program the correct Address Offset for each access.
4-2 —	Reserved
1 COM	Command type This bit indicates the register access type. 1: Indicates a read operation. 0: Indicates a write operation. 0b - Write operation 1b - Read operation
0 OB	Operation Busy. This bit is set along with a read or write command for initiating the indirect access to MAC_<MSEL>_<AOFF> register. This bit is reset when the read or write command to MAC_<MSEL>_<AOFF> register is complete. The next indirect register access can be initiated only after this bit is reset. During a write operation, the bit is reset only after the data has been written into MAC_<MSEL>_<AOFF> register. During a read operation, the data should be read from MAC_Indir_Access_Data register only after this bit is reset.

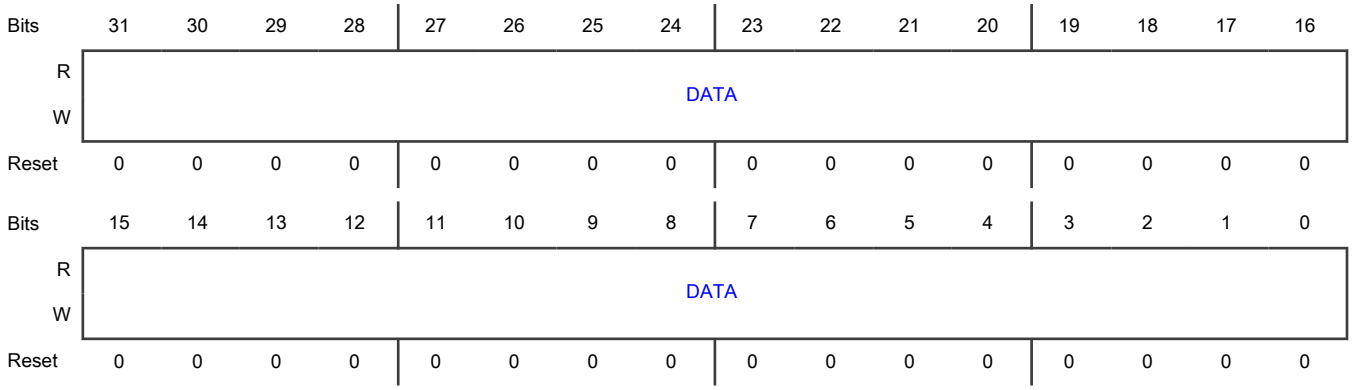
70.5.1.36 Indirect Access Data (INDIR_ACCESS_DATA)

Offset

Register	Offset
INDIR_ACCESS_DATA	A74h

Function
This register holds the read/write data for Indirect Access of MAC registers.

Diagram



Fields

Field	Function
31-0 DATA	This field contains data to read/write for Indirect address access associated with MAC_Indir_Access_Ctrl

70.5.1.37 Timestamp Control (MAC_TIMESTAMP_CONTROL)

Offset

Register	Offset
MAC_TIMESTAMP_CONTROL	B00h

Function
This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the Receiver.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			AV8021A...	Reserved			TXTSSTSM	Reserved			ESTI	Reserved	TSENMAC...	SNAPTYPSEL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TSMSTRE...	TSEVNTE...	TSIPV4E...	TSIPV6E...	TSIPENA	TSVER2E...	TSCTRLS...	TSENALL	Reserved	Reserved	TSADDREG	TSTRIG	TSUPDT	TSINIT	TSCFUPDT	TSENA
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 AV8021ASMEN	AV 802. AV 802.1AS Mode Enable When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS mode of operation. When PTP offload feature is enabled, for the purpose of PTP offload, the transport specific field in the PTP header is generated and checked based on the value of this bit. 0b - AV 802.1AS Mode is disabled 1b - AV 802.1AS Mode is enabled
27-25 —	Reserved
24 TXTSSTSM	Transmit Timestamp Status Mode When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the MAC_Tx_Timestamp_Status_Nanoseconds register. When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the MAC_Tx_Timestamp_Status_Nanoseconds register. 0b - Transmit Timestamp Status Mode is disabled 1b - Transmit Timestamp Status Mode is enabled
23-21 —	Reserved
20	External System Time Input

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>ESTI</td><td>When this bit is set, the MAC uses the external 64-bit reference System Time input for the following:
To take the timestamp provided as status
To insert the timestamp in transmit PTP packets when One-step Timestamp or Timestamp Offload feature is enabled.
When this bit is reset, the MAC uses the internal reference System Time.
0b - External System Time Input is disabled
1b - External System Time Input is enabled</td></tr>
<tr><td>19
—</td><td>Reserved</td></tr>
<tr><td>18
TSENMACADDR</td><td>Enable MAC Address for PTP Packet Filtering
When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.
When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated, when PTP is directly sent over Ethernet.
For normal time stamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.
For PTP offload, only MAC address register 0 is considered for unicast destination address matching.
0b - MAC Address for PTP Packet Filtering is disabled
1b - MAC Address for PTP Packet Filtering is enabled</td></tr>
<tr><td>17-16
SNAPTYPSEL</td><td>Select PTP packets for Taking Snapshots
These bits, along with Bits 15 and 14, decide the set of PTP packet types for which snapshot needs to be taken. The encoding is given in the table below.
<table>
<tr><th>SNAPTYPSEL</th><th>TSMSTRENA</th><th>TSEVNTENA</th><th>PTP Messages</th></tr>
<tr><td>00</td><td>x</td><td>0</td><td>SYNC, Follow_Up, Delay_Req, Delay_Resp</td></tr>
<tr><td>00</td><td>0</td><td>1</td><td>SYNC</td></tr>
<tr><td>00</td><td>1</td><td>1</td><td>Delay_Req</td></tr>
<tr><td>01</td><td>x</td><td>0</td><td>SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up</td></tr>
</table></td></tr>
</table>

Table continued from the previous page...

Field	Function

01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	x	x	SYNC,Pdelay_Req,
11	x	x	Pdelay_Resp Delay_Req

15 TSMSTRENA	Enable Snapshot for Messages Relevant to Master When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node. 0b - Snapshot for Messages Relevant to Master is disabled 1b - Snapshot for Messages Relevant to Master is enabled
14 TSEVNTENA	Enable Timestamp Snapshot for Event Messages When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see Timestamp Snapshot Dependency on Register Bits Table. 0b - Timestamp Snapshot for Event Messages is disabled 1b - Timestamp Snapshot for Event Messages is enabled
13 TSIPV4ENA	Enable Processing of PTP Packets Sent over IPv4-UDP When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default. 0b - Processing of PTP Packets Sent over IPv4-UDP is disabled 1b - Processing of PTP Packets Sent over IPv4-UDP is enabled
12 TSIPV6ENA	Enable Processing of PTP Packets Sent over IPv6-UDP When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets. 0b - Processing of PTP Packets Sent over IPv6-UDP is disabled 1b - Processing of PTP Packets Sent over IPv6-UDP is enabled
11 TSIPENA	Enable Processing of PTP over Ethernet Packets When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets. 0b - Processing of PTP over Ethernet Packets is disabled 1b - Processing of PTP over Ethernet Packets is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 TSVER2ENA	Enable PTP Packet Processing for Version 2 Format Enable PTP Packet Processing for Version 2 Format When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'. 0b - PTP Packet Processing for Version 2 Format is disabled 1b - PTP Packet Processing for Version 2 Format is enabled
9 TSCTRLSSR	Timestamp Digital or Binary Rollover Control When this bit is set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit. 0b - Timestamp Digital Rollover Control is disabled and Binary Rollover Control is enabled 1b - Timestamp Digital Rollover Control is enabled and Binary Rollover Control is disabled
8 TSENALL	Enable Timestamp for All Packets When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC. 0b - Timestamp for All Packets disabled 1b - Timestamp for All Packets enabled
7 —	Reserved
6 —	Reserved
5 TSADDREG	Update Addend Register When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Addend Register is not updated 1b - Addend Register is updated
4 TSTRIG	Enable Timestamp Interrupt Trigger When this bit is set, the timestamp interrupt is generated when the System Time becomes greater than the value written in the Target Time register. This bit is reset after the Timestamp Trigger Interrupt is generated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Timestamp Interrupt Trigger is not enabled 1b - Timestamp Interrupt Trigger is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 TSUPDT	Update Timestamp When this bit is set, the system time is updated (added or subtracted) with the value specified in MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update registers. This bit should be zero before updating it. This bit is reset when the update is complete in hardware. The Timestamp Higher Word register (if enabled while configuring the IP) is not updated. When Media Clock Generation and Recovery is configured (DWC_EQOS_FLEXI_PPS_OUT_EN) and enabled MAC_Presn_Time_Updt should also be updated before setting this field. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Timestamp is not updated 1b - Timestamp is updated
2 TSINIT	Initialize Timestamp When this bit is set, the system time is initialized (overwritten) with the value specified in the MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update registers. This bit should be zero before it is updated. This bit is reset when the initialization is complete. The Timestamp Higher Word register (if enabled while configuration the IP) can only be initialized. When Media Clock Generation and Recovery is configured (DWC_EQOS_FLEXI_PPS_OUT_EN) and enabled MAC_Presn_Time_Updt should also be updated before setting this field. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Timestamp is not initialized 1b - Timestamp is initialized
1 TSCFUPDT	Fine or Coarse Timestamp Update When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp. 0b - Coarse method is used to update system timestamp 1b - Fine method is used to update system timestamp
0 TSENA	Enable Timestamp When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the Receive side, the MAC processes the 1588 packets only if this bit is set. 0b - Timestamp is disabled 1b - Timestamp is enabled

70.5.1.38 Subsecond Increment (MAC_SUB_SECOND_INCREMENT)

Offset

Register	Offset
MAC_SUB_SECOND_INCREMENT	B04h

Function

This register specifies the value to be added to the internal system time register every cycle of clk_ptp_ref_i clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								SNSINC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-16 SNSINC	Sub-nanosecond Increment Value Sub-second Increment Value The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [Bit 9 (TSCTRLSSR) is set in MAC_Timestamp_Control]. When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.
15-8 —	Reserved
7-0 —	Reserved

70.5.1.39 System Time Seconds (MAC_SYSTEM_TIME_SECONDS)

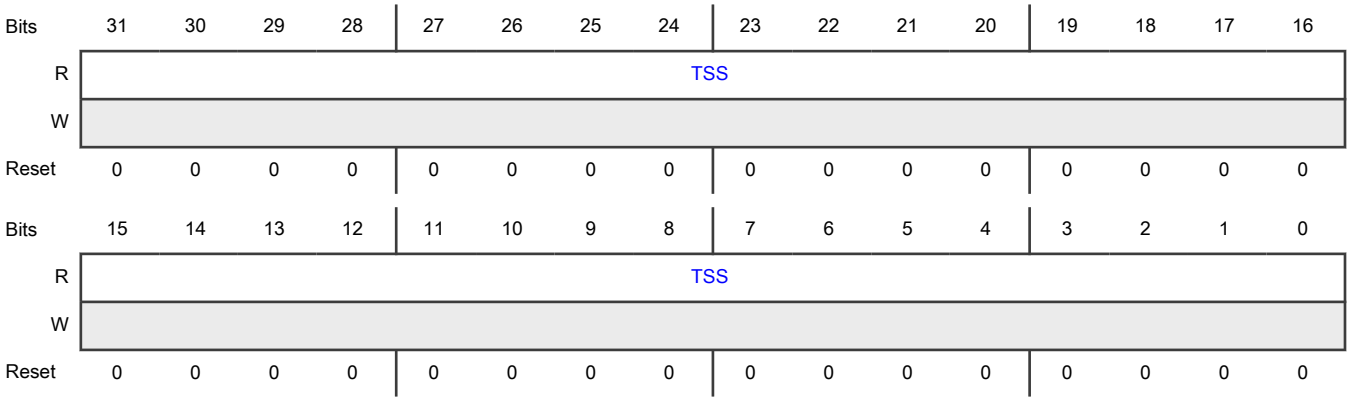
Offset

Register	Offset
MAC_SYSTEM_TIME_SECONDS	B08h

Function

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

Diagram



Fields

Field	Function
31-0 TSS	Timestamp Second The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

70.5.1.40 System Time Nanoseconds (MAC_SYSTEM_TIME_NANOSECONDS)

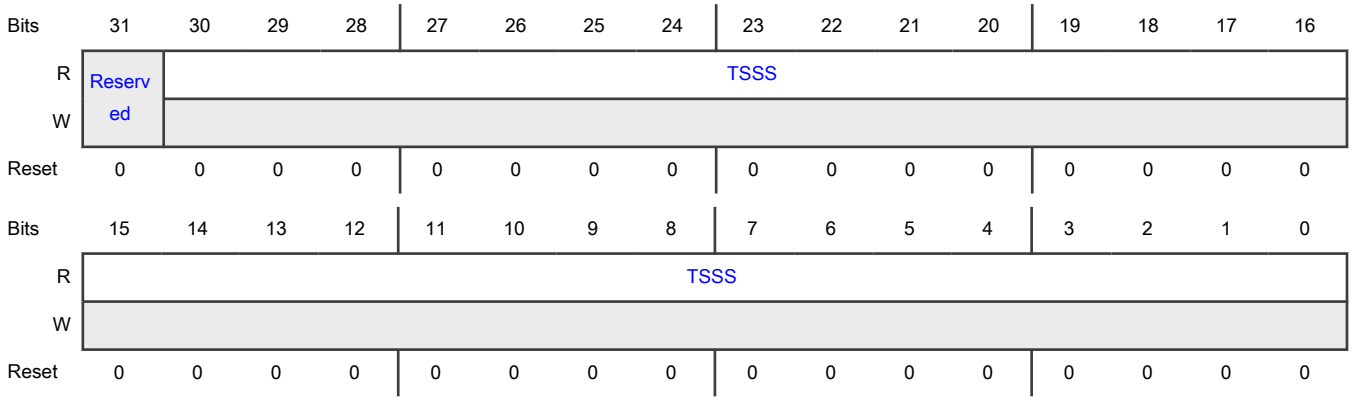
Offset

Register	Offset
MAC_SYSTEM_TIME_NANOSECONDS	B0Ch

Function

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

Diagram



Fields

Field	Function
31 —	Reserved
30-0 TSSS	Timestamp Sub Seconds The value in this field has the sub-second representation of time, with an accuracy of 0. Timestamp Sub Seconds The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When Bit 9 is set in MAC_Timestamp_Control, each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls-over to zero.

70.5.1.41 System Time Seconds Update (MAC_SYSTEM_TIME_SECONDS_UPDATE)

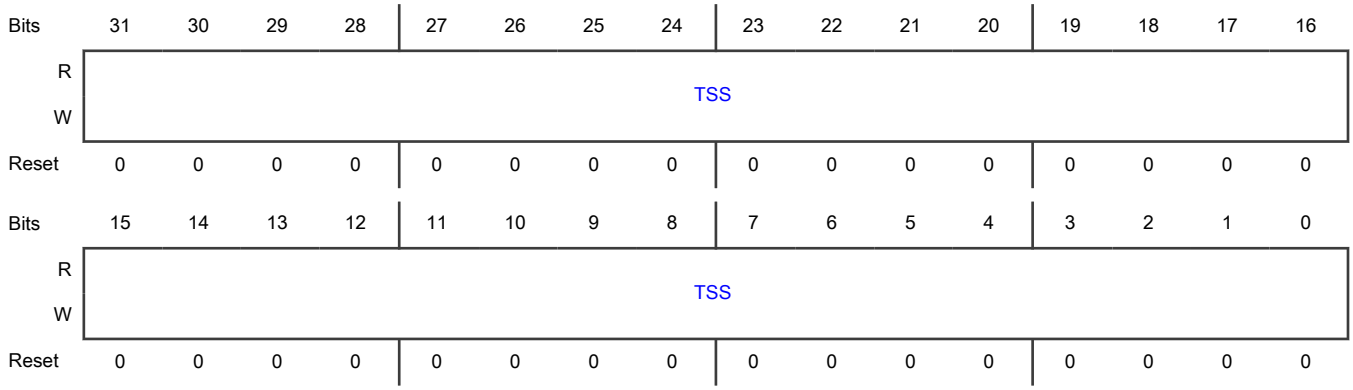
Offset

Register	Offset
MAC_SYSTEM_TIME_SECONDS_UPDATE	B10h

Function

The System Time Seconds Update register, along with the System Time Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in DWC_eqos_top_map/EQOS_MAC/MAC_TIMESTAMP_CONTROL.

Diagram



Fields

Field	Function
31-0 TSS	Timestamp Seconds The value in this field is the seconds part of the update. Timestamp Seconds The value in this field is the seconds part of the update. When ADDSUB is reset, this field must be programmed with the seconds part of the update value. When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value. For example, to subtract 2.000000001 seconds from the system time, the TSS field in the MAC_Timestamp_Seconds_Update register must be 0xFFFF_FFFE (that is, 2^32 - 2).

70.5.1.42 System Time Nanoseconds Update (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)

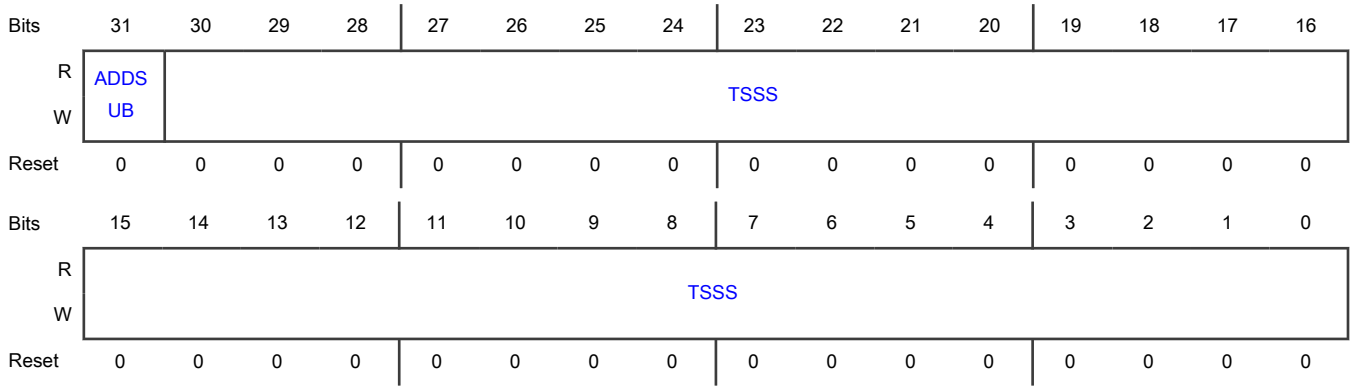
Offset

Register	Offset
MAC_SYSTEM_TIME_NANOSECONDS_UPDATE	B14h

Function

System Time Nanoseconds Update.

Diagram



Fields

Field	Function
31 ADDSUB	Add or Subtract Time Add or Subtract Time When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register. 0b - Add time 1b - Subtract time
30-0 TSSS	Timestamp Sub Seconds Timestamp Sub Seconds The value in this field is the sub-seconds part of the update. ADDSUB is 1: This field must be programmed with the complement of the sub-seconds part of the update value as described. ADDSUB is 0: This field must be programmed with the sub-seconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the MAC_Timestamp_Control register. TSCTRLSSR field in the MAC_Timestamp_Control register is 1: - The programmed value must be 10^9 - <sub-second value>. - Each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF. TSCTRLSSR field in the MAC_Timestamp_Control register is 0: - The programmed value must be 2^31 - <sub-second_value>. - Each bit represents an accuracy of 0.46 ns. For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the MAC_Timestamp_Nanoseconds_Update register must be 0x7FFF_FFFF (that is, 2^31 - 1), when TSCTRLSSR bit in MAC_Timestamp_Control is reset and 0x3B9A_C9FF (that is, 10^9 - 1), when TSCTRLSSR bit in MAC_Timestamp_Control is set.

70.5.1.43 Timestamp Addend (MAC_TIMESTAMP_ADDEND)

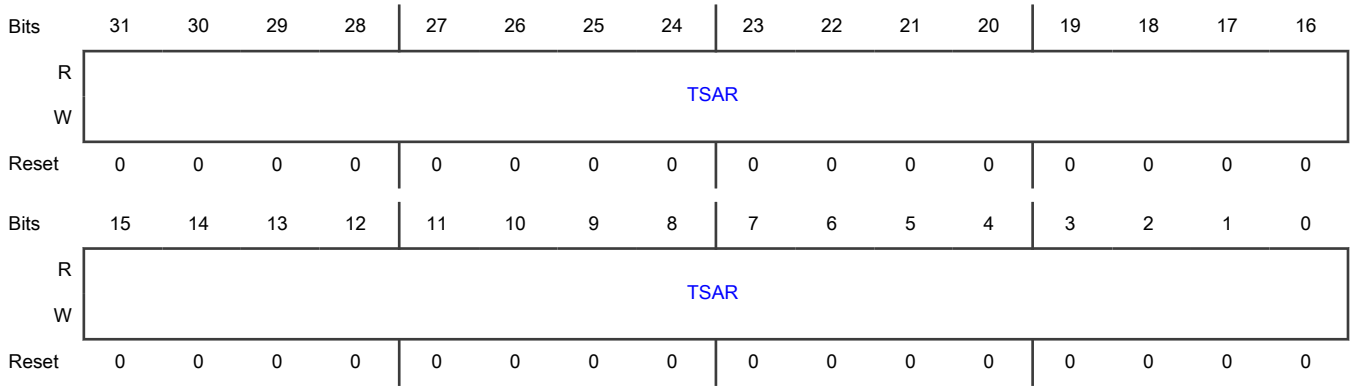
Offset

Register	Offset
MAC_TIMESTAMP_ADDEND	B18h

Function

Timestamp Addend register. This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the MAC_Timestamp_Control register). The content of this register is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated when the accumulator overflows.

Diagram



Fields

Field	Function
31-0 TSAR	Timestamp Addend Register This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

70.5.1.44 Timestamp Status (MAC_TIMESTAMP_STATUS)

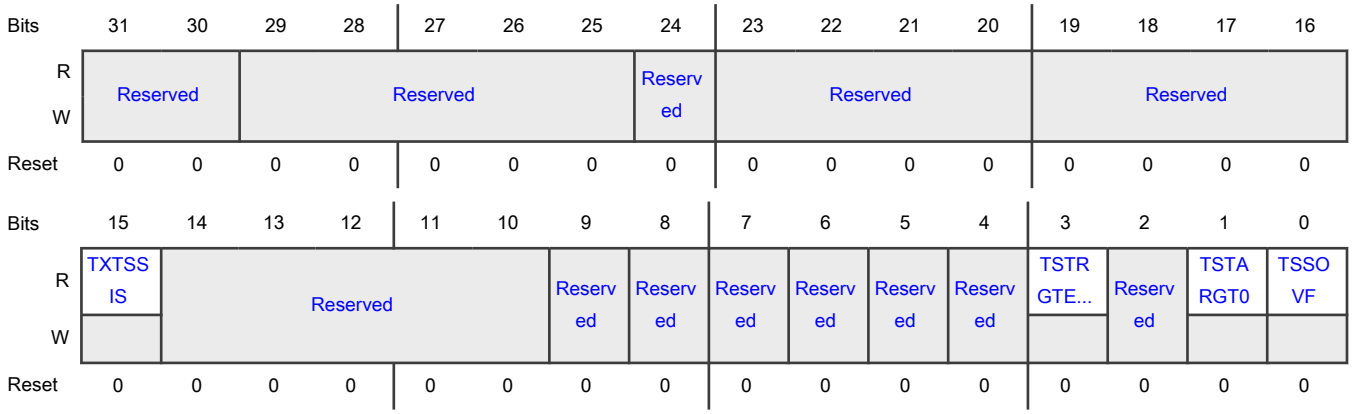
Offset

Register	Offset
MAC_TIMESTAMP_STATUS	B20h

Function

Timestamp Status register. All bits except Bits[27:25] gets cleared when the application reads this register.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-25 —	Reserved
24 —	Reserved
23-20 —	Reserved
19-16 —	Reserved
15 TXTSSIS	Tx Timestamp Status Interrupt Status In non-EQOS_CORE configurations when drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_Tx_Timestamp_Status_Nanoseconds and MAC_Tx_Timestamp_Status_Seconds registers. When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the MAC_Tx_Timestamp_Status_Nanoseconds and MAC_Tx_Timestamp_Status_Seconds registers, for PTO generated Delay Request and Pdelay request packets. This bit is cleared when the MAC_Tx_Timestamp_Status_Seconds register is read (or write to MAC_Tx_Timestamp_Status_Seconds register when RCWE bit of MAC_CSR_SW_Ctrl register is set). 0b - Tx Timestamp Status Interrupt status not detected 1b - Tx Timestamp Status Interrupt status detected
14-10	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved
4 —	Reserved
3 TSTRGTERR0	Timestamp Target Time Error This bit is set when the latest target time programmed in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - Timestamp Target Time Error status not detected 1b - Timestamp Target Time Error status detected
2 —	Reserved
1 TSTARGT0	Timestamp Target Time Reached When this bit is set, it indicates that the value of system time is greater than or equal to the value specified in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - Timestamp Target Time Reached status not detected 1b - Timestamp Target Time Reached status detected
0	Timestamp Seconds Overflow

Table continues on the next page...

Table continued from the previous page...

Field	Function
TSSOVF	When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. 0b - Timestamp Seconds Overflow status not detected 1b - Timestamp Seconds Overflow status detected

70.5.1.45 Transmit Timestamp Status Nanoseconds (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)

Offset

Register	Offset
MAC_TX_TIMESTAMP_STATUS_NANOSECONDS	B30h

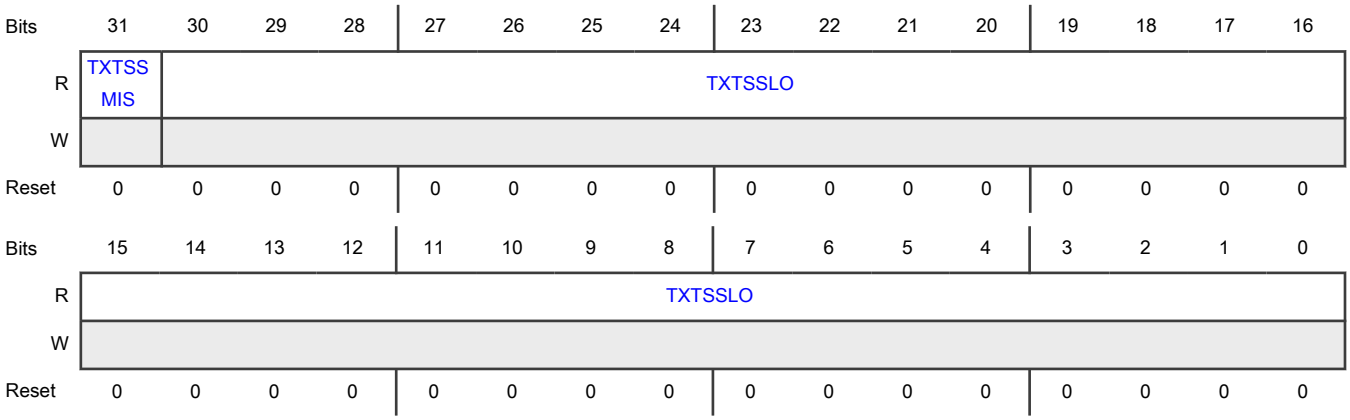
Function

This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.

The MAC_Tx_Timestamp_Status_Nanoseconds register, along with MAC_Tx_Timestamp_Status_Seconds, gives the 64-bit timestamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of MAC_Tx_Timestamp_Status_Nanoseconds is read. In the little-endian mode, this means when bits[31:24] are read; in big-endian mode, bits[7:0] are read.

If the application does not read these registers and timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the setting of the TXTSSTSM bit of the MAC_Timestamp_Control register. The status bit TXTSSIS bit [15] in MAC_Timestamp_Status register is set when the MAC transmitter captures the timestamp.

Diagram



Fields

Field	Function
31 TXTSSMIS	TXTSSMIS Transmit Timestamp Status Missed When this bit is set, it indicates one of the following: The timestamp of the current packet is ignored if TXTSSTSM bit of the MAC_Timestamp_Control register is reset The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the MAC_Timestamp_Control register is set. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Transmit Timestamp Status Missed status not detected 1b - Transmit Timestamp Status Missed status detected
30-0 TXTSSLO	Transmit Timestamp Status Low This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.

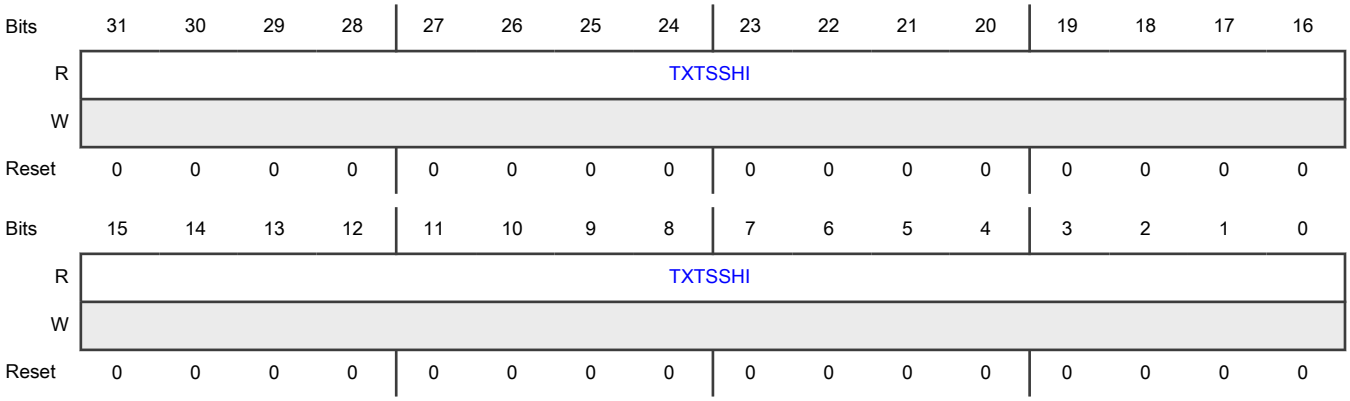
70.5.1.46 Transmit Timestamp Status Seconds (MAC_TX_TIMESTAMP_STATUS_SECONDS)

Offset

Register	Offset
MAC_TX_TIMESTAMP_STATUS_SECONDS	B34h

Function
The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

Diagram



Fields

Field	Function
31-0 TXTSSHI	Transmit Timestamp Status High This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

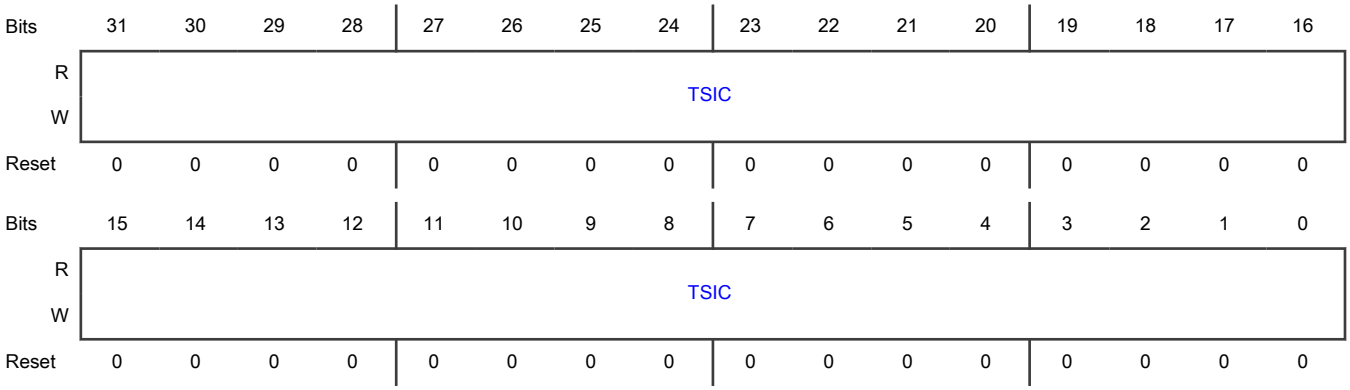
70.5.1.47 Timestamp Ingress Correction Nanosecond (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)

Offset

Register	Offset
MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND	B58h

Function
This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

Diagram



Fields

Field	Function
31-0 TSIC	Timestamp Ingress Correction This field contains the ingress path correction value as defined by the Ingress Correction expression.

70.5.1.48 Timestamp Egress Correction Nanosecond (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)

Offset

Register	Offset
MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND	B5Ch

Function
This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TSEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TSEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 TSEC	Timestamp Egress Correction This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

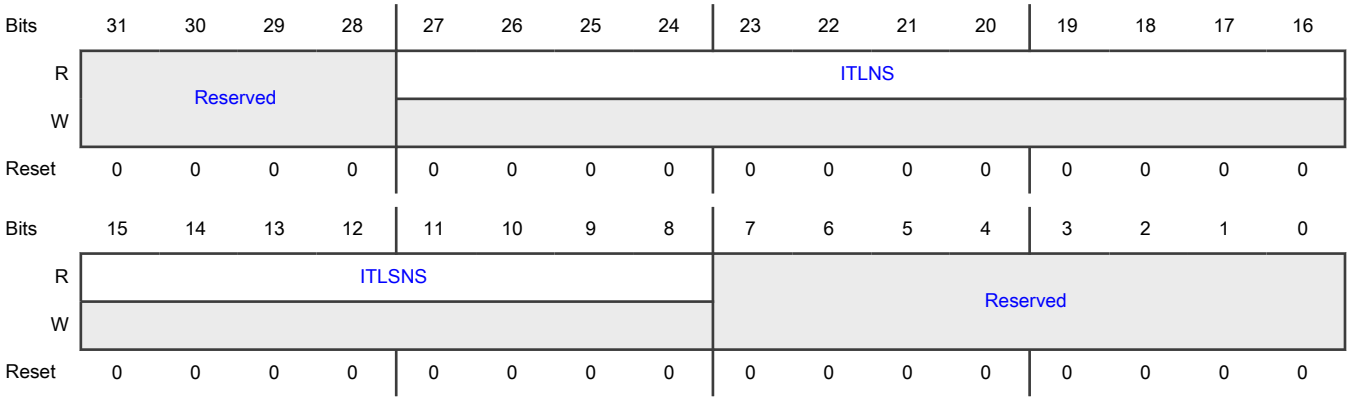
70.5.1.49 Timestamp Ingress Latency (MAC_TIMESTAMP_INGRESS_LATENCY)

Offset

Register	Offset
MAC_TIMESTAMP_INGRESS_LATENCY	B68h

Function
This register holds the Ingress MAC latency.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-16 ITLNS	ITLNS Ingress Timestamp Latency, in sub-nanoseconds This register holds the average latency in sub-nanoseconds between the input ports (phy_rxd_i) of MAC and the actual point (GMII/MII) where the ingress timestamp is taken. Ingress correction value is computed as described in the section Timestamp Correction.
15-8 ITLSNS	ITLSNS Ingress Timestamp Latency, in nanoseconds This register holds the average latency in nanoseconds between the input ports (phy_rxd_i) of MAC and the actual point (GMII/MII) where the ingress timestamp is taken. Ingress correction value is computed as described in the section, "Ingress Correction".
7-0 —	Reserved

70.5.1.50 Timestamp Egress Latency (MAC_TIMESTAMP_EGRESS_LATENCY)

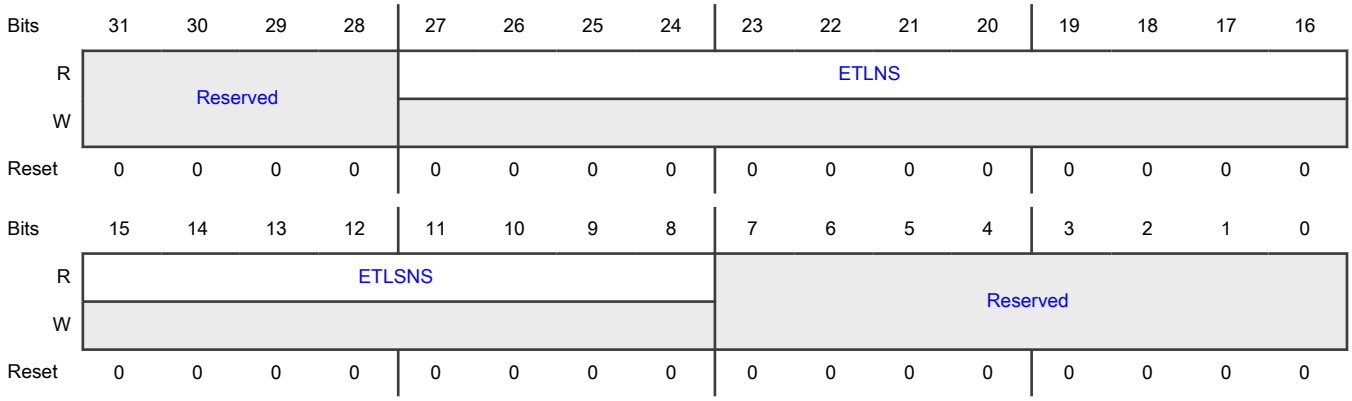
Offset

Register	Offset
MAC_TIMESTAMP_EGRESS_LATENCY	B6Ch

Function

This register holds the Egress MAC latency.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-16 ETLNS	Egress Timestamp Latency, in nanoseconds This register holds the average latency in nanoseconds between the actual point (GMII/MII) where the egress timestamp is taken and the output ports (phy_txd_o) of the MAC. Ingress correction value is computed as described in the section Timestamp correction..
15-8 ETLSNS	Egress Timestamp Latency, in sub-nanoseconds This register holds the average latency in sub-nanoseconds between the actual point (GMII/MII) where the egress timestamp is taken and the output ports (phy_txd_o) of the MAC. Ingress correction value is computed as described in the section Timestamp correction.
7-0 —	Reserved

70.5.1.51 PPS Control (MAC_PPS_CONTROL)

Offset

Register	Offset
MAC_PPS_CONTROL	B70h

Function

Bits[30:24] of this register are valid only when four Flexible PPS outputs are selected. Bits[22:16] are valid only when three or more Flexible PPS outputs are selected. Bits[14:8] are valid only when two or more Flexible PPS outputs are selected. Bits[6:4] are valid only when Flexible PPS feature is selected.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	Reserved		Reserved	Reserved				Reserved	Reserved		Reserved	Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	Reserved		Reserved	Reserved				Reserved	Reserved		Reserved	PPSCTRL_PPSCMD			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 —	Reserved
27-24 —	Reserved
23 —	Reserved
22-21 —	Reserved
20 —	Reserved
19-16 —	Reserved
15 —	Reserved
14-13 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 —	Reserved
11-8 —	Reserved
7 —	Reserved
6-5 —	Reserved
4 —	Reserved
3-0 PPSCTRL_PPSCMD	PPS Output Frequency Control This field controls the frequency of the PPS0 output (ptp_pps_o[0]) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies: 0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz. 0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz. 0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz. 0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz. .. 1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz. Note: In the binary rollover mode, the PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example: When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of One clock of 50 percent duty cycle and 537 ms period Second clock of 463 ms period (268 ms low and 195 ms high) When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of Three clocks of 50 percent duty cycle and 268 ms period Fourth clock of 195 ms period (134 ms low and 61 ms high)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This behavior is because of the non-linear toggling of bits in the digital rollover mode in the MAC_System_Time_Nanoseconds register. or Flexible PPS Output (ptp_pps_o[0]) Control Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are 'all-zero'. The following list describes the values of PPSCMD0: 0000: No Command 0001: START Single Pulse This command generates single pulse rising at the start point defined in MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds register and of a duration defined in the PPS0 Width Register. 0010: START Pulse Train This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS0 Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands. 0011: Cancel START This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time. 0100: STOP Pulse train at time This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programmed in the Target Time registers elapses. 0101: STOP Pulse Train immediately This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010). 110: Cancel STOP Pulse train This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command. 0111-1111: Reserved

70.5.1.52 PPS0 Target Time Seconds (PPS0_TARGET_TIME_SECONDS)

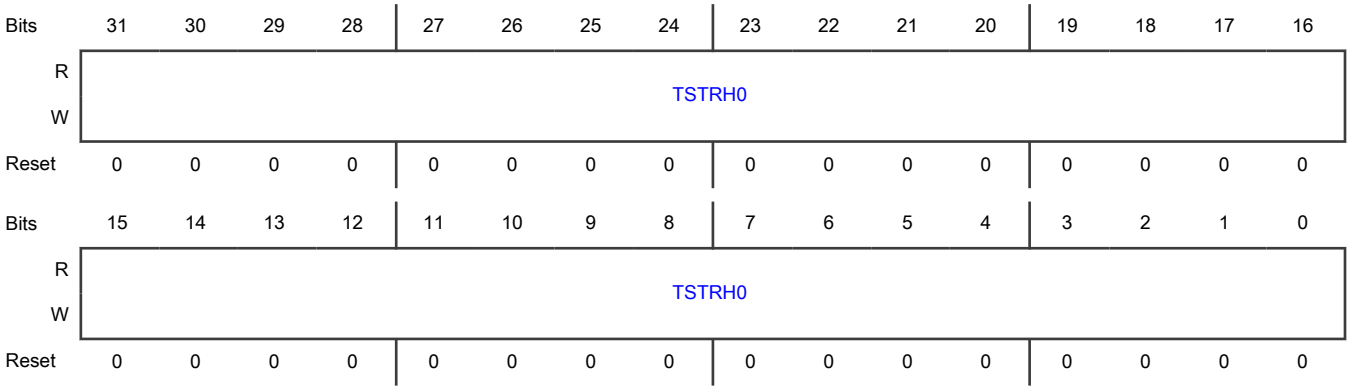
Offset

Register	Offset
PPS0_TARGET_TIME_SECONDS	B80h

Function

The PPS Target Time Seconds register, along with PPS Target Time Nanoseconds register, is used to schedule an interrupt event [Bit 1 of MAC_Timestamp_Status] when the system time exceeds the value programmed in these registers.

Diagram



Fields

Field	Function
31-0 TSTRH0	PPS Target Time Seconds Register PPS Target Time Seconds Register This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the MAC_PPS_Control register. If DWC_EQOS_FLEXI_PPS_OUT_EN is enabled in the configuration and PTGE field of MAC_Timestamp_Control Register is set with Presentation time control set in recovery mode, then these bits indicate the TPT being programmed by the application and in generation mode it indicates the CPT generated at the sampled trigger.

70.5.1.53 PPS0 Target Time Nanoseconds (PPS0_TARGET_TIME_NANOSECONDS)

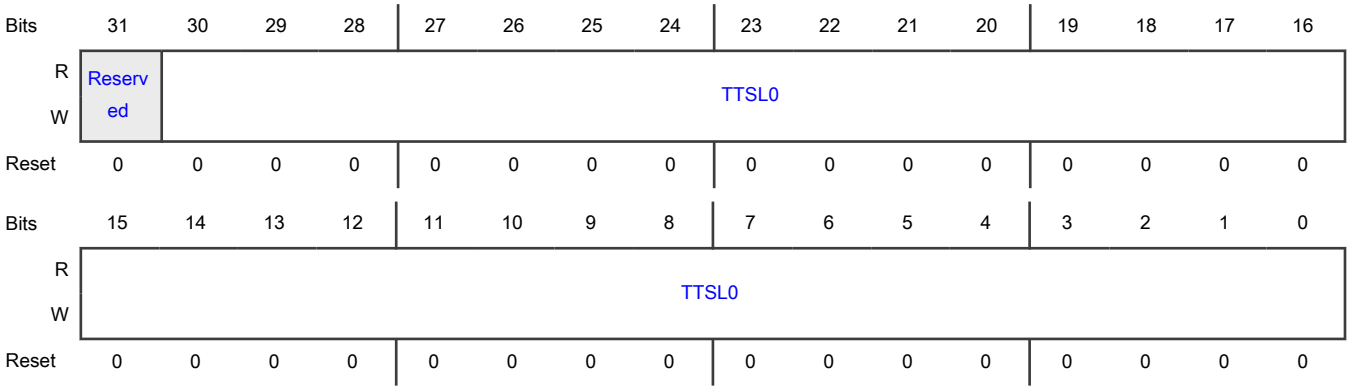
Offset

Register	Offset
PPS0_TARGET_TIME_NANOSECONDS	B84h

Function

PPS0 Target Time Nanoseconds.

Diagram



Fields

Field	Function
31 —	Reserved
30-0 TTSL0	Target Time Low for PPS Register Target Time Low for PPS Register This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in MAC_PPS_Control. When the TSCTRLSSR bit is reset in the MAC_Timestamp_Control register, this value should be (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of sub-second increment value. When the TSCTRLSSR bit is set in the MAC_Timestamp_Control register, this value should not exceed 0x3B9A_C9FF. The actual start or stop time of the PPS signal output might have an error margin up to one unit of sub-second increment value. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

70.5.1.54 MTL Operation Mode (MTL_OPERATION_MODE)

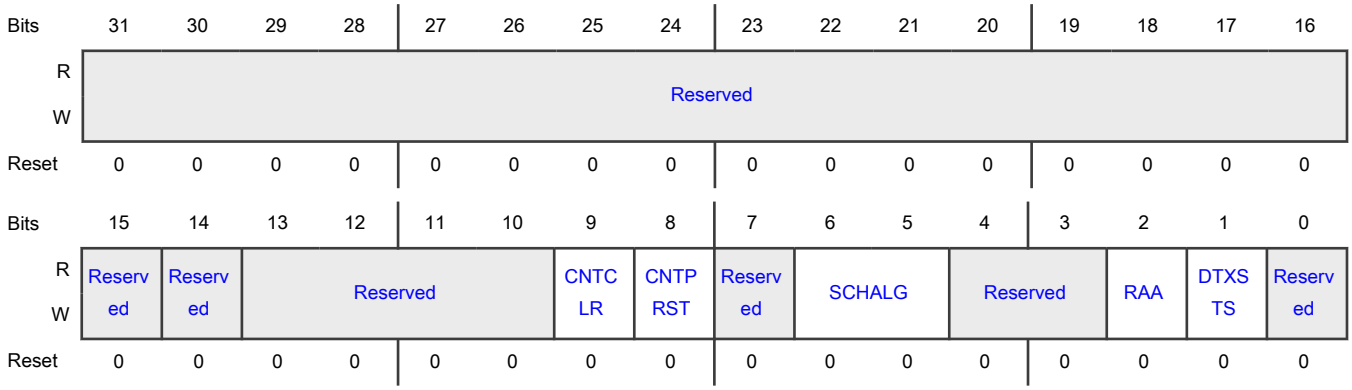
Offset

Register	Offset
MTL_OPERATION_MODE	C00h

Function

The Operation Mode register establishes the Transmit and Receive operating modes and commands.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 —	Reserved
14 —	Reserved
13-10 —	Reserved
9 CNTCLR	Counters Reset When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNT_PRESET bit, CNT_PRESET has precedence. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Counters are not reset 1b - All counters are reset
8 CNTPRST	Counters Preset When this bit is set, MTL_TxQ[0-7]_Underflow register is initialized/preset to 12'h7F0. Missed Packet and Overflow Packet counters in MTL_RxQ[0-7]_Missed_Packet_Overflow_Cnt register is initialized/preset to 12'h7F0. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Counters Preset is disabled 1b - Counters Preset is enabled
7	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
6-5 SCHALG	Tx Scheduling Algorithm This field indicates the algorithm for Tx scheduling: 00b - WRR algorithm 01b - WFQ algorithm when DCB feature is selected.Otherwise, Reserved 10b - DWRR algorithm when DCB feature is selected.Otherwise, Reserved 11b - Strict priority algorithm
4-3 —	Reserved
2 RAA	Receive Arbitration Algorithm This field is used to select the arbitration algorithm for the Rx side. 0: Strict priority (SP). Queue 0 has the lowest priority and the last queue has the highest priority. 1: Weighted Strict Priority (WSP) 0b - Strict priority (SP) 1b - Weighted Strict Priority (WSP)
1 DTXSTS	Drop Transmit Status 1: Tx packet status received from the MAC is dropped in the MTL 0: Tx packet status received from the MAC is forwarded to the application 0b - Drop Transmit Status is disabled 1b - Drop Transmit Status is enabled
0 —	Reserved

70.5.1.55 MTL Interrupt Status (MTL_INTERRUPT_STATUS)

Offset

Register	Offset
MTL_INTERRUPT_STATUS	C20h

Function

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								Reserved	Reserved				Reserved	Reserved	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Q1IS	Q0IS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23 —	Reserved
22-19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15-8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 —	Reserved
3 —	Reserved
2 —	Reserved
1 Q1IS	Queue 1 Interrupt status This bit indicates that there is an interrupt from Queue 1. To reset this bit, the application must read the MTL_Q1_Interrupt_Control_Status register to get the exact cause of the interrupt and clear its source. 0b - Queue 1 Interrupt status not detected 1b - Queue 1 Interrupt status detected
0 Q0IS	Queue 0 Interrupt status This bit indicates that there is an interrupt from Queue 0. To reset this bit, the application must read Queue 0 Interrupt Control and Status register to get the exact cause of the interrupt and clear its source. 0b - Queue 0 Interrupt status not detected 1b - Queue 0 Interrupt status detected

70.5.1.56 Receive Queue and DMA Channel Mapping 0 (MTL_RXQ_DMA_MAP0)

Offset

Register	Offset
MTL_RXQ_DMA_MAP0	C30h

Function

The Receive Queue and DMA Channel Mapping 0 register is reserved in EQOS-CORE and EQOS-MTL configurations.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			Reserved	Reserved			Reserved	Reserved			Reserved	Reserved			Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			Q1DDMACH	Reserved			Q1MDMACH	Reserved			Q0DDMACH	Reserved			Q0MDMACH
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 —	Reserved
27-25 —	Reserved
24 —	Reserved
23-21 —	Reserved
20 —	Reserved
19-17 —	Reserved
16 —	Reserved
15-13 —	Reserved
12 Q1DDMACH	Queue 1 Enabled for DA-based DMA Channel Selection

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When set, this bit indicates that the packets received in Queue 1 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 1 are routed to the DMA Channel programmed in the Q1MDMACH field (Bits[10:8]). 0b - Queue 1 disabled for DA-based DMA Channel Selection 1b - Queue 1 enabled for DA-based DMA Channel Selection
11-9 —	Reserved
8 Q1MDMACH	Queue 1 Mapped to DMA Channel This field controls the routing of the received packet in Queue 1 to the DMA channel: 0: DMA Channel 0 1: DMA Channel 1 This field is valid when Q1DDMACH field is reset.
7-5 —	Reserved
4 Q0DDMACH	Queue 0 Enabled for DA-based DMA Channel Selection When set, this bit indicates that the packets received in Queue 0 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 0 are routed to the DMA Channel programmed in the Q0MDMACH field. 0b - Queue 0 disabled for DA-based DMA Channel Selection 1b - Queue 0 enabled for DA-based DMA Channel Selection
3-1 —	Reserved
0 Q0MDMACH	Queue 0 Mapped to DMA Channel This field controls the routing of the packet received in Queue 0 to the DMA channel: 0: DMA Channel 0 1: DMA Channel 1 This field is valid when Q0DDMACH field is reset.

70.5.1.57 Queue 0 Transmit Operation Mode (MTL_TXQ0_OPERATION_MODE)

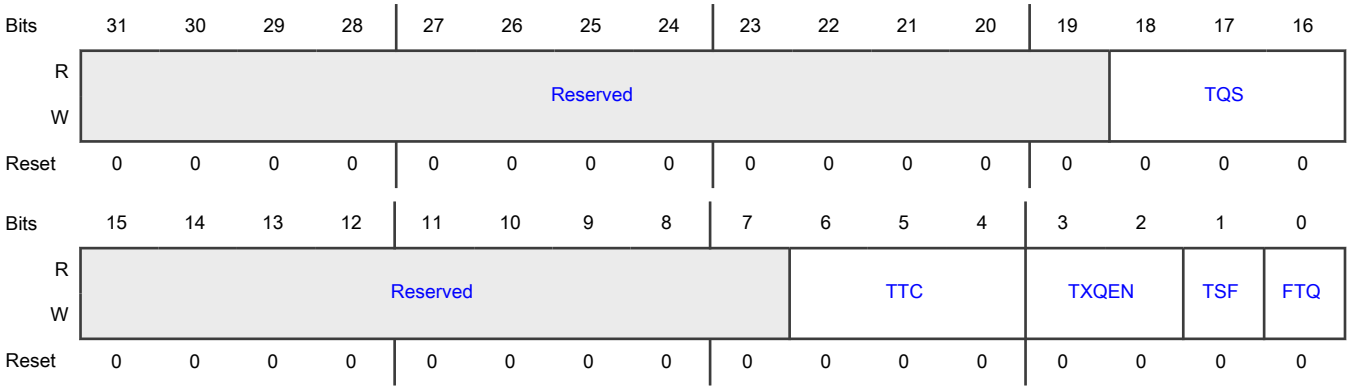
Offset

Register	Offset
MTL_TXQ0_OPERATION_MODE	D00h

Function

The Queue 0 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

Diagram



Fields

Field	Function
31-19 —	Reserved
18-16 TQS	Transmit Queue Size This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The TQS field is read-write only if the number of Tx Queues more than one, the reset value is 0x0 and indicates size of 256 bytes. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the Queue = (TQS+1)*256 bytes. When the number of Tx Queues is one, the field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits
15-7 —	Reserved
6-4	Transmit Threshold Control

Table continues on the next page...

Table continued from the previous page...

Field	Function
TTC	These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. 000b - 32 001b - 64 010b - 96 011b - 128 100b - 192 101b - 256 110b - 384 111b - 512
3-2 TXQEN	Transmit Queue Enable This field is used to enable/disable the transmit queue 0. 2'b00: Not enabled 2'b01: Reserved 2'b10: Enabled 2'b11: Reserved This field is Read Only in Single Queue configurations and Read Write in Multiple Queue configurations. Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field. 00b - Not enabled 01b - Enable in AV mode (Reserved in non-AV) 10b - Enabled 11b - Reserved
1 TSF	Transmit Store and Forward When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped. 0b - Transmit Store and Forward is disabled 1b - Transmit Store and Forward is enabled
0 FTQ	Flush Transmit Queue When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted

Table continues on the next page...

Table continued from the previous page...

Field	Function
	by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Flush Transmit Queue is disabled 1b - Flush Transmit Queue is enabled

70.5.1.58 Queue 0 Underflow Counter (MTL_TXQ0_UNDERFLOW)

Offset

Register	Offset
MTL_TXQ0_UNDERFLOW	D04h

Function

The Queue 0 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				UFCNTOVF	UFFRMCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 UFCNTOVF	Overflow Bit for Underflow Packet Counter This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Overflow not detected for Underflow Packet Counter 1b - Overflow detected for Underflow Packet Counter
10-0 UFFRMCNT	Underflow Packet Counter This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

70.5.1.59 Queue 0 Transmit Debug (MTL_TXQ0_DEBUG)

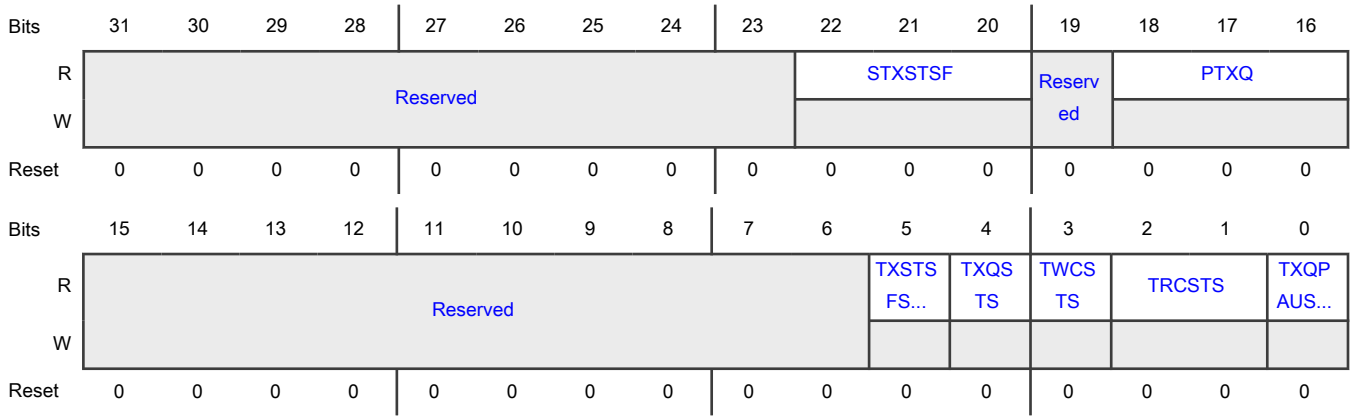
Offset

Register	Offset
MTL_TXQ0_DEBUG	D08h

Function

The Queue 0 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-20 STXSTSF	Number of Status Words in Tx Status FIFO of Queue This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.
19 —	Reserved
18-16 PTXQ	Number of Packets in the Transmit Queue This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue.
15-6 —	Reserved
5 TXSTSFSTS	MTL Tx Status FIFO Full Status When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission. 0b - MTL Tx Status FIFO Full status is not detected 1b - MTL Tx Status FIFO Full status is detected
4 TXQSTS	MTL Tx Queue Not Empty Status When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission. 0b - MTL Tx Queue Not Empty status is not detected 1b - MTL Tx Queue Not Empty status is detected
3 TWCSTS	MTL Tx Queue Write Controller Status When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue. 0b - MTL Tx Queue Write Controller status is not detected 1b - MTL Tx Queue Write Controller status is detected
2-1 TRCSTS	MTL Tx Queue Read Controller Status This field indicates the state of the Tx Queue Read Controller: 00b - Idle state 01b - Read state (transferring data to the MAC transmitter)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Waiting for pending Tx Status from the MAC transmitter 11b - Flushing the Tx queue because of the Packet Abort request from the MAC
0 TXQPAUSED	Transmit Queue in Pause When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled Reception of 802.3x Pause packet when PFC is disabled 0b - Transmit Queue in Pause status is not detected 1b - Transmit Queue in Pause status is detected

70.5.1.60 Queue 0 ETS Status (MTL_TXQ0_ETS_STATUS)

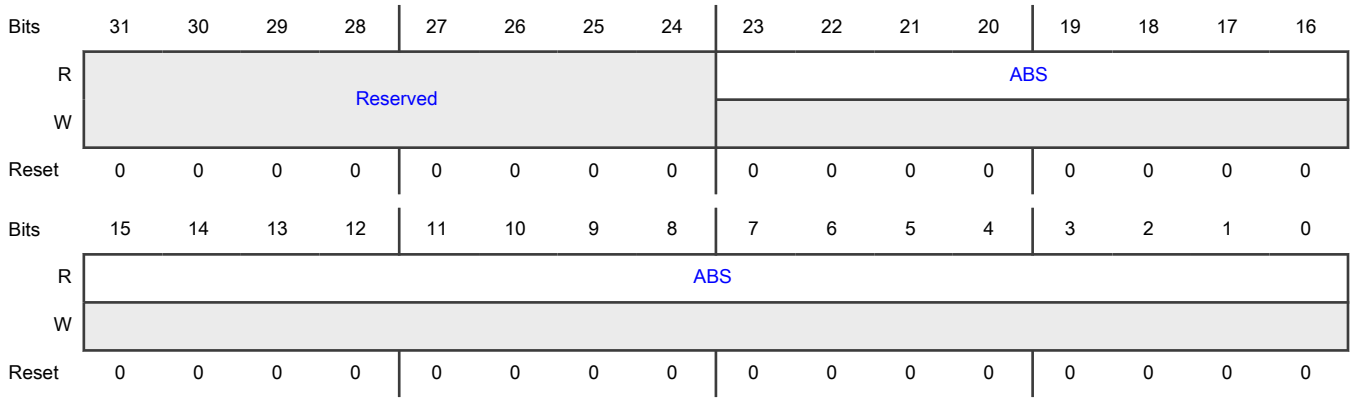
Offset

Register	Offset
MTL_TXQ0_ETS_STATUS	D14h

Function

The Queue 0 ETS Status register provides the average traffic transmitted in Queue 0.

Diagram



Fields

Field	Function
31-24	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
23-0 ABS	Average Bits per Slot This field contains the average transmitted bits per slot. When the DCB operation is enabled for Queue 0, this field is computed over every 10 million bit times slot (4 ms in 2500 Mbps; 10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680.

70.5.1.61 Queue 0 Quantum or Weights (MTL_TXQ0_QUANTUM_WEIGHT)

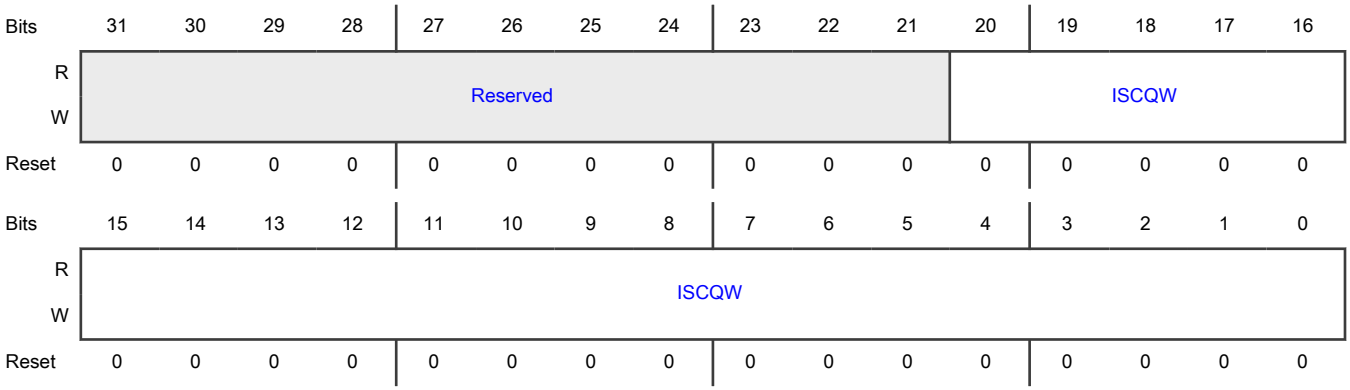
Offset

Register	Offset
MTL_TXQ0_QUANTUM_WEIGHT	D18h

Function

The Queue 0 Quantum or Weights register contains the quantum value for Deficit Weighted Round Robin (DWRR), weights for the Weighted Round Robin (WRR), and Weighted Fair Queuing (WFQ) for Queue 0.

Diagram



Fields

Field	Function
31-21 —	Reserved
20-0 ISCQW	Quantum or Weights

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When the DCB operation is enabled with DWRR algorithm for Queue 0 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. When DCB operation is enabled with WFQ algorithm for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. The higher the programmed weights lesser the bandwidth allocated for the particular Transmit Queue. This is because the weights are used to compute the packet finish time (weights * packet_size). Lesser the finish time, higher the probability of the packet getting scheduled first and using more bandwidth. When DCB operation or generic queuing operation is enabled with WRR algorithm for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero.

70.5.1.62 Queue 0 Interrupt Control Status (MTL_Q0_INTERRUPT_CONTROL_STATUS)

Offset

Register	Offset
MTL_Q0_INTERRUPT_CONTROL_STATUS	D2Ch

Function

This register contains the interrupt enable and status bits for the queue 0 interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							RXOIE	Reserved							RXOVFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved						ABPSIE	TXUIE	Reserved						ABPSIS	TXUNFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
24 RXOIE	Receive Queue Overflow Interrupt Enable When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled. 0b - Receive Queue Overflow Interrupt is disabled 1b - Receive Queue Overflow Interrupt is enabled
23-17 —	Reserved
16 RXOVFIS	Receive Queue Overflow Interrupt Status This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Queue Overflow Interrupt Status not detected 1b - Receive Queue Overflow Interrupt Status detected
15-10 —	Reserved
9 ABPSIE	Average Bits Per Slot Interrupt Enable When this bit is set, the MAC asserts the sbd_intr_o or mci_intr_o interrupt when the average bits per slot status is updated. When this bit is cleared, the interrupt is not asserted for such an event. 0b - Average Bits Per Slot Interrupt is disabled 1b - Average Bits Per Slot Interrupt is enabled
8 TXUIE	Transmit Queue Underflow Interrupt Enable When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled. 0b - Transmit Queue Underflow Interrupt Status is disabled 1b - Transmit Queue Underflow Interrupt Status is enabled
7-2 —	Reserved
1 ABPSIS	Average Bits Per Slot Interrupt Status When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Average Bits Per Slot Interrupt Status not detected 1b - Average Bits Per Slot Interrupt Status detected
0 TXUNFIS	Transmit Queue Underflow Interrupt Status This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Queue Underflow Interrupt Status not detected 1b - Transmit Queue Underflow Interrupt Status detected

70.5.1.63 Queue 0 Receive Operation Mode (MTL_RXQ0_OPERATION_MODE)

Offset

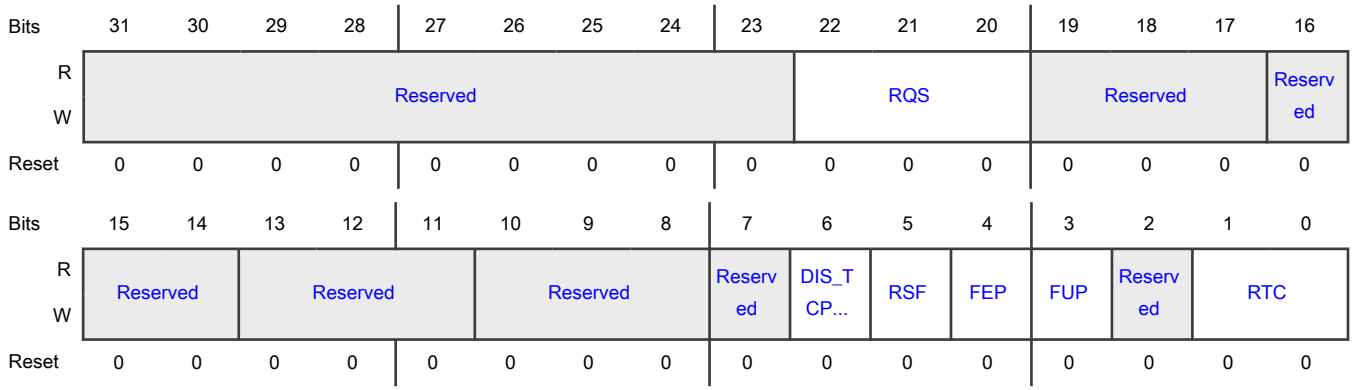
Register	Offset
MTL_RXQ0_OPERATION_MODE	D30h

Function

The Queue 0 Receive Operation Mode register establishes the Receive queue operating modes and command.

The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release

Diagram



Fields

Field	Function
31-23 —	Reserved
22-20 RQS	Receive Queue Size This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one, the reset value is 0x0 and indicates size of 256 bytes. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the Queue = (RQS+1)*256 bytes. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits
19-17 —	Reserved
16-14 —	Reserved
13-11 —	Reserved
10-8 —	Reserved
7 —	Reserved
6 DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset. 0b - Dropping of TCP/IP Checksum Error Packets is enabled 1b - Dropping of TCP/IP Checksum Error Packets is disabled
5 RSF	Receive Queue Store and Forward When this bit is set, the enet reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Receive Queue Store and Forward is disabled 1b - Receive Queue Store and Forward is enabled
4 FEP	Forward Error Packets When this bit is reset, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet might be forwarded to the application or DMA. 0b - Forward Error Packets is disabled 1b - Forward Error Packets is enabled
3 FUP	Forward Undersized Good Packets When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01. 0b - Forward Undersized Good Packets is disabled 1b - Forward Undersized Good Packets is enabled
2 —	Reserved
1-0 RTC	Receive Queue Threshold Control These bits control the threshold level of the MTL Rx queue (in bytes): The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1. 00b - 64 01b - 32 10b - 96 11b - 128

70.5.1.64 Queue 0 Missed Packet and Overflow Counter (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)

Offset

Register	Offset
MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT	D34h

Function

The Queue 0 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				MISCNTO...	MISPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				OVFCNTO...	OVFPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27 MISCNTOVF	Missed Packet Counter Overflow Bit When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Missed Packet Counter overflow not detected 1b - Missed Packet Counter overflow detected
26-16 MISPKTCNT	Missed Packet Counter This field indicates the number of packets missed by the enet because the application asserted ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Access restriction applies. Clears on read. Self-set to 1 on internal event.
15-12 —	Reserved
11 OVFCNTOVF	Overflow Counter Overflow Bit When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Overflow Counter overflow not detected 1b - Overflow Counter overflow detected
10-0 OVFPKTCNT	Overflow Packet Counter This field indicates the number of packets discarded by the enet because of Receive queue overflow. This counter is incremented each time the enet discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

70.5.1.65 Queue 0 Receive Debug (MTL_RXQ0_DEBUG)

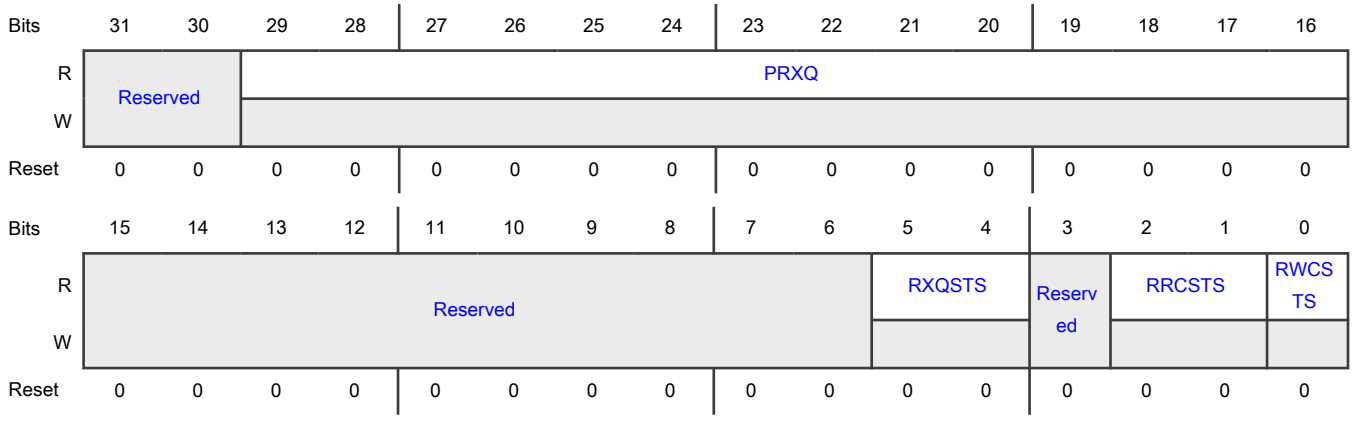
Offset

Register	Offset
MTL_RXQ0_DEBUG	D38h

Function

The Queue 0 Receive Debug register gives the debug status of various blocks related to the Receive queue.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-16 PRXQ	Number of Packets in Receive Queue This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.
15-6 —	Reserved
5-4 RXQSTS	MTL Rx Queue Fill-Level Status This field gives the status of the fill-level of the Rx Queue: 00b - Rx Queue empty 01b - Rx Queue fill-level below flow-control deactivate threshold 10b - Rx Queue fill-level above flow-control activate threshold 11b - Rx Queue full
3 —	Reserved
2-1 RRCSTS	MTL Rx Queue Read Controller State This field gives the state of the Rx queue Read controller: 00b - Idle state 01b - Reading packet data 10b - Reading packet status (or timestamp) 11b - Flushing the packet data and status
0 RWCSTS	MTL Rx Queue Write Controller Active Status When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue. 0b - MTL Rx Queue Write Controller Active Status not detected 1b - MTL Rx Queue Write Controller Active Status detected

70.5.1.66 Queue 0 Receive Control (MTL_RXQ0_CONTROL)

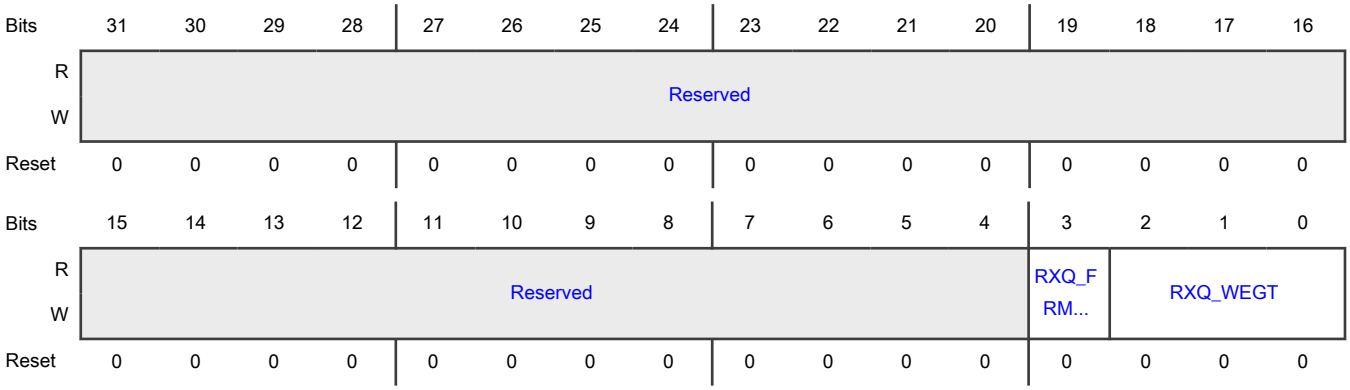
Offset

Register	Offset
MTL_RXQ0_CONTROL	D3Ch

Function

The Queue 0 Receive Control register controls the receive arbitration and passing of received packets to the application.

Diagram



Fields

Field	Function
31-4 —	Reserved
3 RXQ_FRM_ARBIT	Receive Queue Packet Arbitration When this bit is set, the enet drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the enet drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: PBL amount of data (indicated by ari_qN_pbl_i[]) or Complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the enet drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode). 0b - Receive Queue Packet Arbitration is disabled 1b - Receive Queue Packet Arbitration is enabled
2-0 RXQ_WEGT	Receive Queue Weight This field indicates the weight assigned to the Rx Queue 0. This field needs to be programmed with one value less than the required weight, that is, reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the MTL_RxQn_Control registers must be programmed before the MTL_Operation_Mode register.

70.5.1.67 Queue 1 Transmit Operation Mode (MTL_TXQ1_OPERATION_MODE)

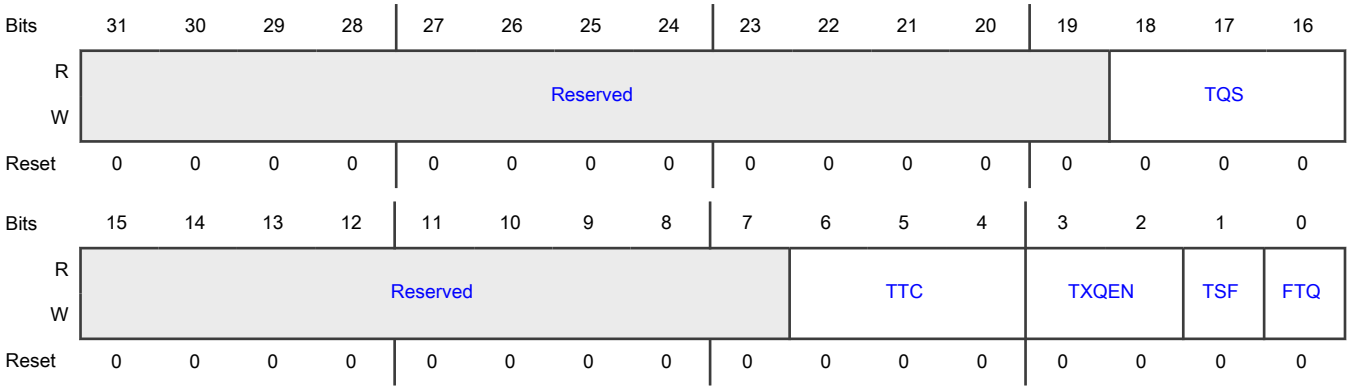
Offset

Register	Offset
MTL_TXQ1_OPERATION_MODE	D40h

Function

The Queue 1 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

Diagram



Fields

Field	Function
31-19 —	Reserved
18-16 TQS	Transmit Queue Size This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The TQS field is read-write only if the number of Tx Queues more than one, the reset value is 0x0 and indicates size of 256 bytes. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the Queue = (TQS+1)*256 bytes. When the number of Tx Queues is one, the field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits
15-7 —	Reserved
6-4	Transmit Threshold Control

Table continues on the next page...

Table continued from the previous page...

Field	Function
TTC	These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. 000b - 32 001b - 64 010b - 96 011b - 128 100b - 192 101b - 256 110b - 384 111b - 512
3-2 TXQEN	Transmit Queue Enable This field is used to enable/disable the transmit queue 0. 2'b00: Not enabled 2'b01: Enable in AV mode 2'b10: Enabled 2'b11: Reserved Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field. 00b - Not enabled 01b - Enable in AV mode (Reserved in non-AV) 10b - Enabled 11b - Reserved
1 TSF	Transmit Store and Forward When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped. 0b - Transmit Store and Forward is disabled 1b - Transmit Store and Forward is enabled
0 FTQ	Flush Transmit Queue When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Flush Transmit Queue is disabled 1b - Flush Transmit Queue is enabled

70.5.1.68 Queue 1 Underflow Counter (MTL_TXQ1_UNDERFLOW)

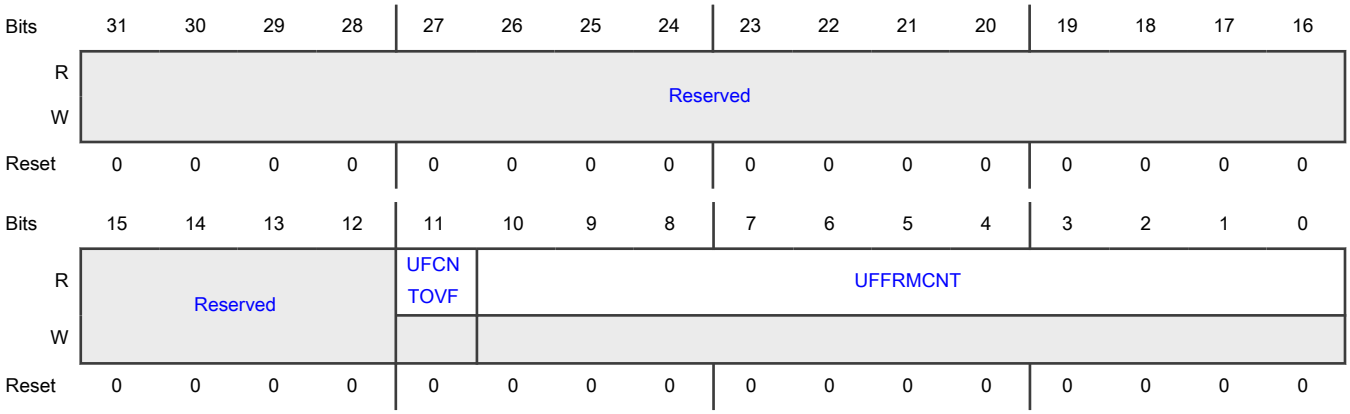
Offset

Register	Offset
MTL_TXQ1_UNDERFLOW	D44h

Function

The Queue 1 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

Diagram



Fields

Field	Function
31-12 —	Reserved
11	Overflow Bit for Underflow Packet Counter

Table continues on the next page...

Table continued from the previous page...

Field	Function
UFCNTOVF	This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Overflow not detected for Underflow Packet Counter 1b - Overflow detected for Underflow Packet Counter
10-0 UFFRMCNT	Underflow Packet Counter This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

70.5.1.69 Queue 1 Transmit Debug (MTL_TXQ1_DEBUG)

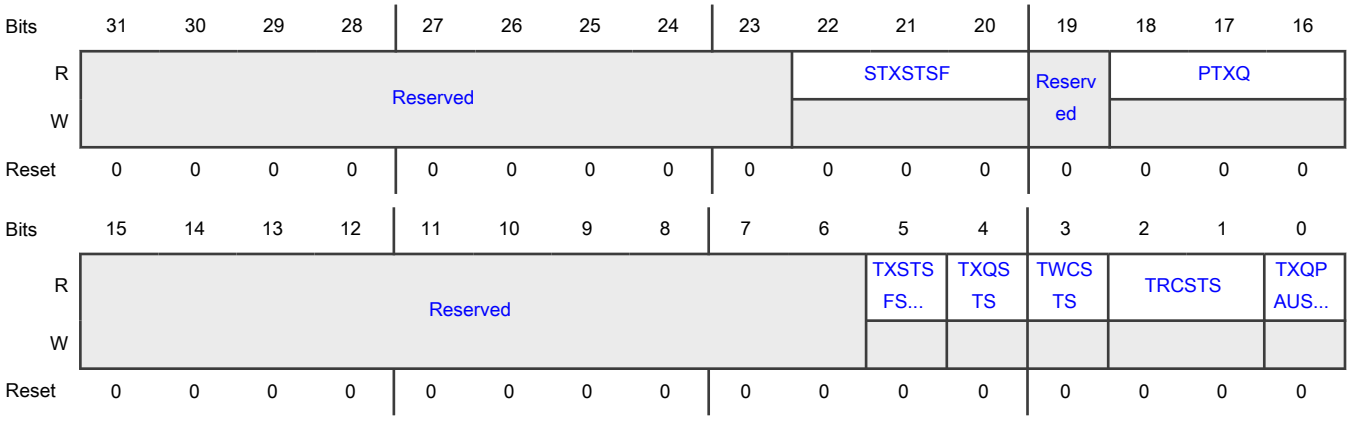
Offset

Register	Offset
MTL_TXQ1_DEBUG	D48h

Function

The Queue 1 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-20 STXSTSF	Number of Status Words in Tx Status FIFO of Queue This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.
19 —	Reserved
18-16 PTXQ	Number of Packets in the Transmit Queue This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue.
15-6 —	Reserved
5 TXSTSFSTS	MTL Tx Status FIFO Full Status MTL Tx Status FIFO Full Status When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission. 0b - MTL Tx Status FIFO Full status is not detected 1b - MTL Tx Status FIFO Full status is detected
4 TXQSTS	MTL Tx Queue Not Empty Status When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission. 0b - MTL Tx Queue Not Empty status is not detected 1b - MTL Tx Queue Not Empty status is detected
3 TWCSTS	MTL Tx Queue Write Controller Status When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue. 0b - MTL Tx Queue Write Controller status is not detected 1b - MTL Tx Queue Write Controller status is detected
2-1 TRCSTS	MTL Tx Queue Read Controller Status This field indicates the state of the Tx Queue Read Controller: 00b - Idle state

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Read state (transferring data to the MAC transmitter) 10b - Waiting for pending Tx Status from the MAC transmitter 11b - Flushing the Tx queue because of the Packet Abort request from the MAC
0 TXQPAUSED	Transmit Queue in Pause When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled Reception of 802.3x Pause packet when PFC is disabled 0b - Transmit Queue in Pause status is not detected 1b - Transmit Queue in Pause status is detected

70.5.1.70 Queue 1 ETS Control (MTL_TXQ1_ETS_CONTROL)

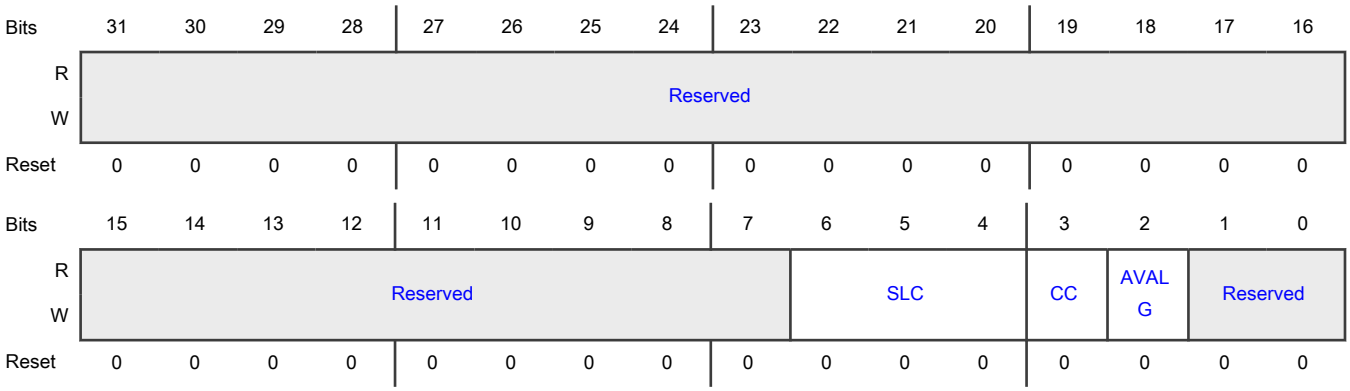
Offset

Register	Offset
MTL_TXQ1_ETS_CONTROL	D50h

Function

The Queue 1 ETS Control register controls the enhanced transmission selection operation.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-4 SLC	Slot Count If the credit-based shaper algorithm is enabled, the software can program the number of slots (of duration programmed in DMA_CH(#i)_Slot_Interval register) over which the average transmitted bits per slot, provided in the MTL_TxQ(#i)_ETS_Status register, need to be computed for Queue. The encoding is as follows: 000b - 1 slot 001b - 2 slots 010b - 4 slots 011b - 8 slots 100b - 16 slots 101b - Reserved
3 CC	Credit Control When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in Channel 1. The credit accumulates even when there is no packet waiting in Channel 1 and another channel is transmitting. When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in Channel 1. When there is no packet waiting in Channel 1 and other channel is transmitting, no credit is accumulated. 0b - Credit Control is disabled 1b - Credit Control is enabled
2 AVALG	AV Algorithm When Queue 1 is programmed for AV, this field configures the scheduling algorithm for this queue: This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue 1 traffic. When reset, strict priority is selected. 0b - CBS Algorithm is disabled 1b - CBS Algorithm is enabled
1-0 —	Reserved

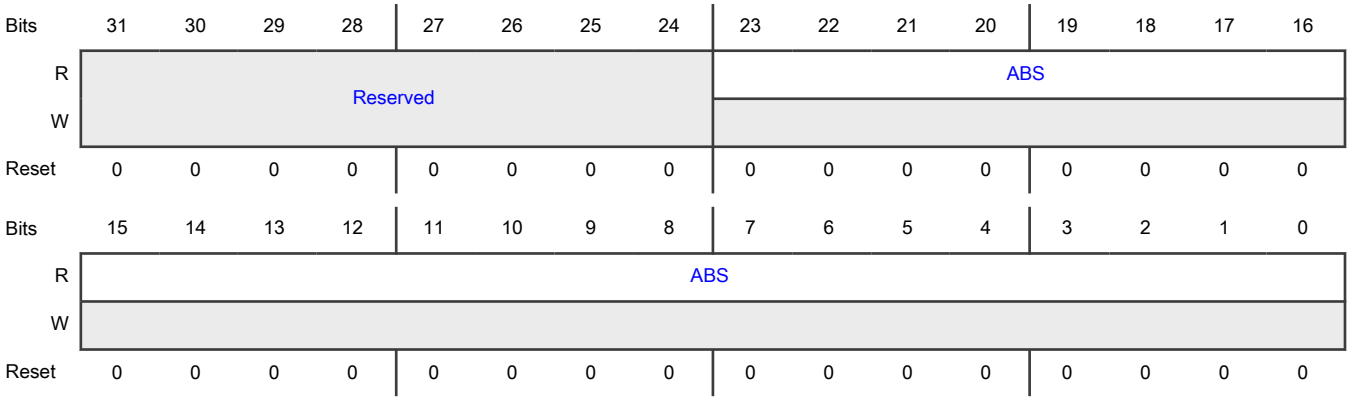
70.5.1.71 Queue 1 ETS Status (MTL_TXQ1_ETS_STATUS)

Offset

Register	Offset
MTL_TXQ1_ETS_STATUS	D54h

Function
The Queue 1 ETS Status register provides the average traffic transmitted in Queue 1.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 ABS	Average Bits per Slot This field contains the average transmitted bits per slot. If AV operation is enabled, this field is computed over number of slots, programmed in the SLC field of MTL_TxQ(#i)_ETS_CONTROL register. The maximum value of this field is 0x6_4000 in 100 Mbps, 0x3E_8000 in 1000 Mbps and 9C_4000 in 2500 Mbps mode respectively. When the DCB operation is enabled for Queue, this field is computed over every 10 million bit times slot (4 ms in 2500 Mbps; 10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680.

70.5.1.72 Queue 1 idleSlopeCredit, Quantum or Weights (MTL_TXQ1_QUANTUM_WEIGHT)

Offset

Register	Offset
MTL_TXQ1_QUANTUM_WEIGHT	D58h

Function

The Queue 1 idleSlopeCredit, Quantum or Weights register provides the average traffic transmitted in Queue 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												ISCQW			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISCQW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved
20-0 ISCQW	idleSlopeCredit, Quantum or Weights When AV feature is enabled, this field contains the idleSlopeCredit value required for the credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps; 3.2 ns for 2500 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbps mode and 0x1000 in 100 Mbps mode. Bits[20:14] must be written to zero. Quantum When the DCB operation is enabled with DWRR algorithm for Queue 1 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. Weights When DCB operation is enabled with WFQ algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. When DCB operation or generic queuing operation is enabled with WRR algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero. Note 1: In multiple Queue configuration this field in respective per queue register must be programmed to some non-zero value when multiple queues are enabled or single queue other than Q0 is enabled. This field need not be programmed when only Q0 is enabled. In general, when WRR algorithm is selected a non-zero value must be programmed on both Receive and Transmit. In Receive, the register is MTL_Operation_Mode register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Note 2: For WFQ algorithm, higher the programmed weights lesser the bandwidth allocated for that Transmit Queue. The finish time is not a function of particular packet alone but it is as per the formula: (previous_finish_time of particular Transmit Queue + (weights*packet_size)) Note 3: The weights programmed do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue w.r.t. total BW or time.

70.5.1.73 Queue 1 sendSlopeCredit (MTL_TXQ1_SENDSLOPECREDIT)

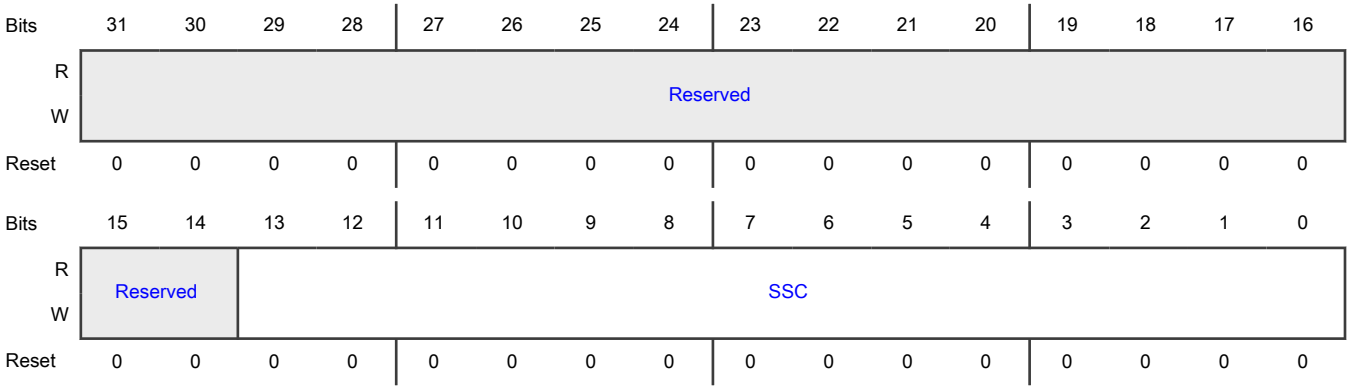
Offset

Register	Offset
MTL_TXQ1_SENDSLOPECREDIT	D5Ch

Function

The sendSlopeCredit register contains the sendSlope credit value required for the credit-based shaper algorithm for the Queue.

Diagram



Fields

Field	Function
31-14 —	Reserved
13-0 SSC	sendSlopeCredit Value When AV operation is enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns, 8 ns and 3.2 ns for 100 Mbps, 1000 Mbps and 2500 Mbps respectively) when the credit is decreasing. The software

Table continues on the next page...

Table continued from the previous page...

Field	Function
	should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbps mode and 0x1000 in 100 Mbps mode. This field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when Channel 1 is selected for transmission.

70.5.1.74 Queue 1 hiCredit (MTL_TXQ1_HICREDIT)

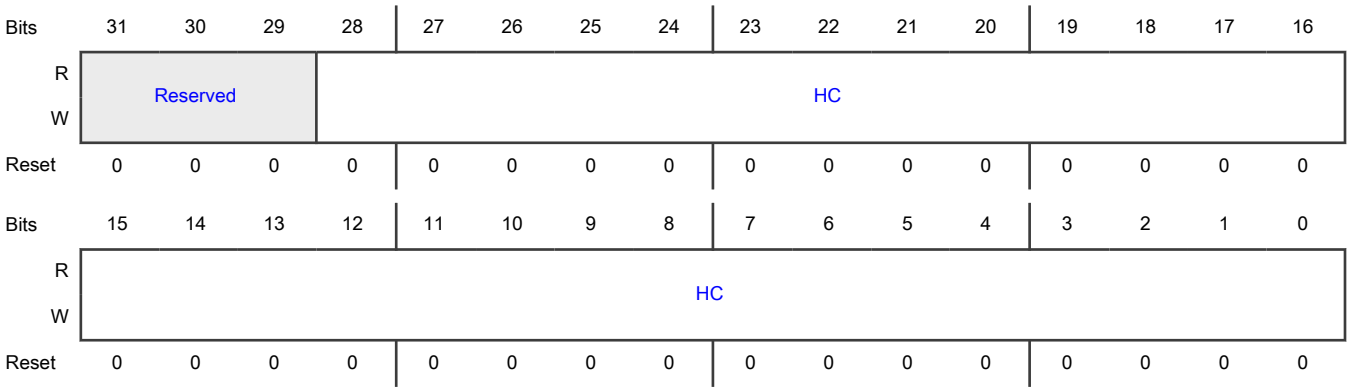
Offset

Register	Offset
MTL_TXQ1_HICREDIT	D60h

Function

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the Queue.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-0 HC	hiCredit Value When the AV feature is enabled, this field contains the hiCredit value required for the credit-based shaper algorithm. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The value to be specified is 131,072 * 1,024 = 134,217,728 or 0x0800_0000.

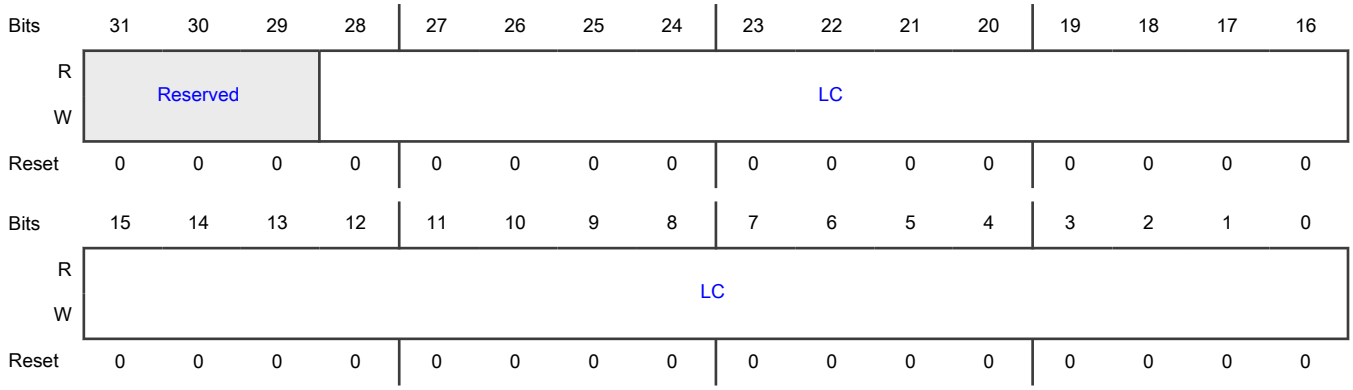
70.5.1.75 Queue 1 loCredit (MTL_TXQ1_LOCREDIT)

Offset

Register	Offset
MTL_TXQ1_LOCREDIT	D64h

Function
The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the Queue.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-0 LC	loCredit Value When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value to be programmed is corresponds to twice the maxFrameSize transmitted from this queue. If the maxFrameSize is 8192 bytes, then (8192*2) * 8 * 1024 = 134,217,728 or 0x0800_0000. Because it is a negative value, the programmed value is 2's complement of the value, that is, 0x1800_0000.

70.5.1.76 Queue 1 Interrupt Control Status (MTL_Q1_INTERRUPT_CONTROL_STATUS)

Offset

Register	Offset
MTL_Q1_INTERRUPT_CONTROL_STATUS	D6Ch

Function

This register contains the interrupt enable and status bits for the queue 1 interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							RXOIE	Reserved							RXOVFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved						ABPSIE	TXUIE	Reserved						ABPSIS	TXUNFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 —	Reserved
24 RXOIE	Receive Queue Overflow Interrupt Enable When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled. 0b - Receive Queue Overflow Interrupt is disabled 1b - Receive Queue Overflow Interrupt is enabled
23-17 —	Reserved
16 RXOVFIS	Receive Queue Overflow Interrupt Status This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Queue Overflow Interrupt Status not detected 1b - Receive Queue Overflow Interrupt Status detected
15-10 —	Reserved
9 ABPSIE	Average Bits Per Slot Interrupt Enable When this bit is set, the MAC asserts the sbd_intr_o or mci_intr_o interrupt when the average bits per slot status is updated.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When this bit is cleared, the interrupt is not asserted for such an event. 0b - Average Bits Per Slot Interrupt is disabled 1b - Average Bits Per Slot Interrupt is enabled
8 TXUIE	Transmit Queue Underflow Interrupt Enable When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled. 0b - Transmit Queue Underflow Interrupt Status is disabled 1b - Transmit Queue Underflow Interrupt Status is enabled
7-2 —	Reserved
1 ABPSIS	Average Bits Per Slot Interrupt Status When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Average Bits Per Slot Interrupt Status not detected 1b - Average Bits Per Slot Interrupt Status detected
0 TXUNFIS	Transmit Queue Underflow Interrupt Status This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Queue Underflow Interrupt Status not detected 1b - Transmit Queue Underflow Interrupt Status detected

70.5.1.77 Queue 1 Receive Operation Mode (MTL_RXQ1_OPERATION_MODE)

Offset

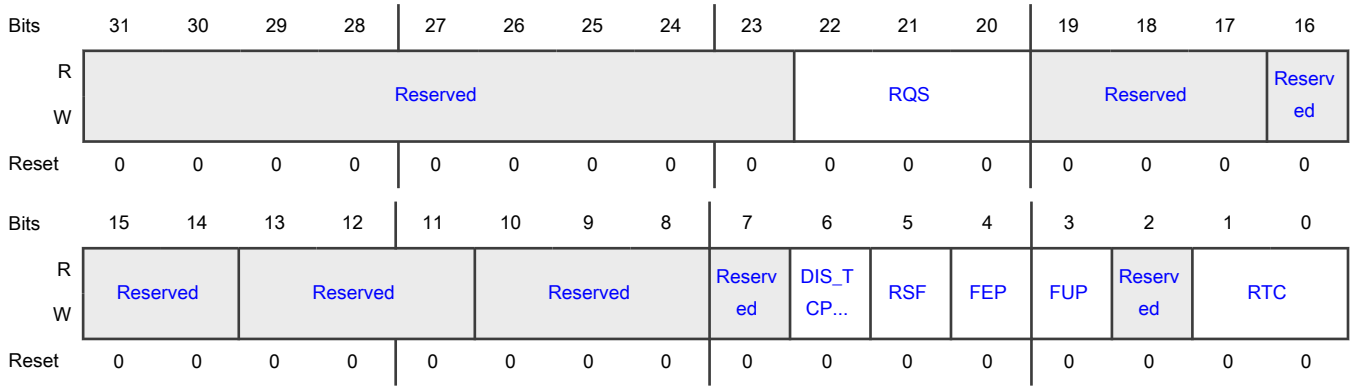
Register	Offset
MTL_RXQ1_OPERATION_MODE	D70h

Function

The Queue 1 Receive Operation Mode register establishes the Receive queue operating modes and command.

The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release

Diagram



Fields

Field	Function
31-23 —	Reserved
22-20 RQS	Receive Queue Size Receive Queue Size This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one, the reset value is 0x0 and indicates size of 256 bytes. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the Queue = (RQS+1)*256 bytes. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits
19-17 —	Reserved
16-14 —	Reserved
13-11 —	Reserved
10-8 —	Reserved
7 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset. 0b - Dropping of TCP/IP Checksum Error Packets is enabled 1b - Dropping of TCP/IP Checksum Error Packets is disabled
5 RSF	Receive Queue Store and Forward When this bit is set, the enet reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register. 0b - Receive Queue Store and Forward is disabled 1b - Receive Queue Store and Forward is enabled
4 FEP	Forward Error Packets When this bit is reset, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet might be forwarded to the application or DMA. 0b - Forward Error Packets is disabled 1b - Forward Error Packets is enabled
3 FUP	Forward Undersized Good Packets When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01. 0b - Forward Undersized Good Packets is disabled 1b - Forward Undersized Good Packets is enabled
2 —	Reserved
1-0 RTC	Receive Queue Threshold Control These bits control the threshold level of the MTL Rx queue (in bytes):

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1. 00b - 64 01b - 32 10b - 96 11b - 128

70.5.1.78 Queue 1 Missed Packet and Overflow Counter (MTL_RXQ1_MISSED_PACKET_OVERFLOW_CNT)

Offset

Register	Offset
MTL_RXQ1_MISSED_PACKET_OVERFLOW_CNT	D74h

Function

The Queue 1 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				MISCNTO...	MISPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				OVFCNTO...	OVFPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
27 MISCNTOVF	Missed Packet Counter Overflow Bit When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Missed Packet Counter overflow not detected 1b - Missed Packet Counter overflow detected
26-16 MISPKTCNT	Missed Packet Counter This field indicates the number of packets missed by the enet because the application asserted ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.
15-12 —	Reserved
11 OVFCNTOVF	Overflow Counter Overflow Bit When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Overflow Counter overflow not detected 1b - Overflow Counter overflow detected
10-0 OVFPKTCNT	Overflow Packet Counter This field indicates the number of packets discarded by the enet because of Receive queue overflow. This counter is incremented each time the enet discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

70.5.1.79 Queue 1 Receive Debug (MTL_RXQ1_DEBUG)

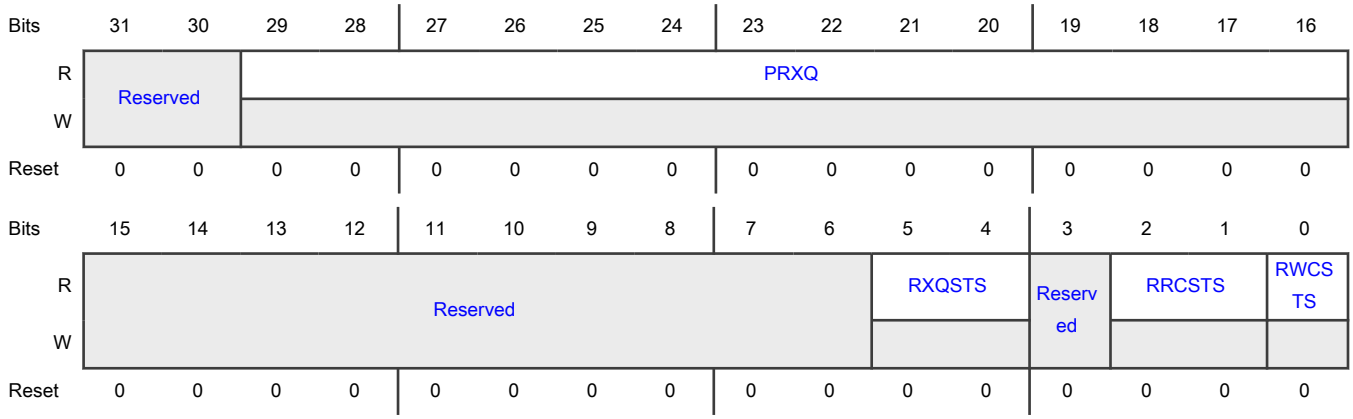
Offset

Register	Offset
MTL_RXQ1_DEBUG	D78h

Function

The Queue 1 Receive Debug register gives the debug status of various blocks related to the Receive queue.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-16 PRXQ	Number of Packets in Receive Queue This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.
15-6 —	Reserved
5-4 RXQSTS	MTL Rx Queue Fill-Level Status This field gives the status of the fill-level of the Rx Queue: 00b - Rx Queue empty 01b - Rx Queue fill-level below flow-control deactivate threshold 10b - Rx Queue fill-level above flow-control activate threshold 11b - Rx Queue full
3 —	Reserved
2-1 RRCSTS	MTL Rx Queue Read Controller State This field gives the state of the Rx queue Read controller: 00b - Idle state 01b - Reading packet data 10b - Reading packet status (or timestamp) 11b - Flushing the packet data and status

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 RWCSTS	MTL Rx Queue Write Controller Active Status When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue. 0b - MTL Rx Queue Write Controller Active Status not detected 1b - MTL Rx Queue Write Controller Active Status detected

70.5.1.80 Queue 1 Receive Control (MTL_RXQ1_CONTROL)

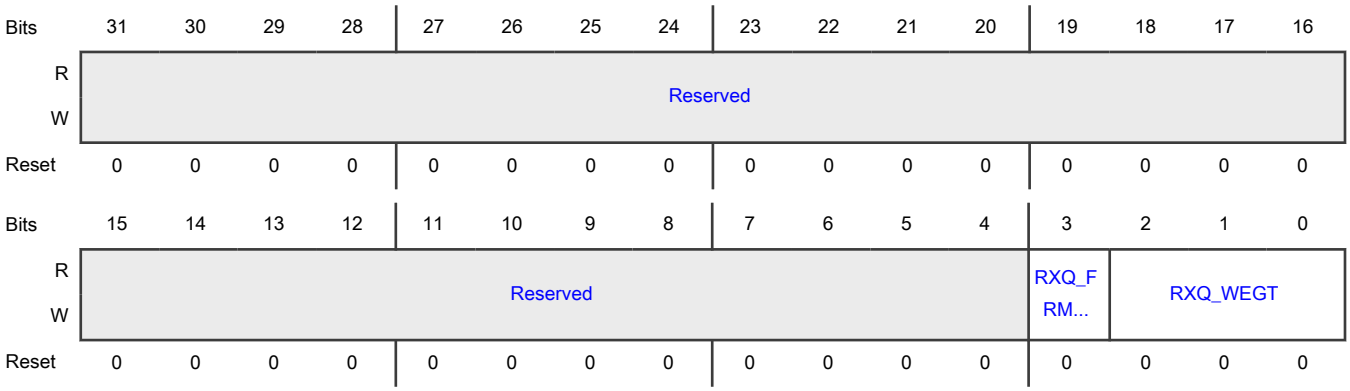
Offset

Register	Offset
MTL_RXQ1_CONTROL	D7Ch

Function

The Queue 1 Receive Control register controls the receive arbitration and passing of received packets to the application.

Diagram



Fields

Field	Function
31-4 —	Reserved
3 RXQ_FRM_ARBIT	Receive Queue Packet Arbitration When this bit is set, the enet drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the enet drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	PBL amount of data (indicated by ari_qN_pbl_i[]) or Complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the enet drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode). 0b - Receive Queue Packet Arbitration is disabled 1b - Receive Queue Packet Arbitration is enabled
2-0 RXQ_WEGT	Receive Queue Weight This field indicates the weight assigned to the Rx Queue 0. This field needs to be programmed with one value less than the required weight, that is, reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the MTL_RxQn_Control registers must be programmed before the MTL_Operation_Mode register.

70.5.1.81 DMA Bus Mode (DMA_MODE)

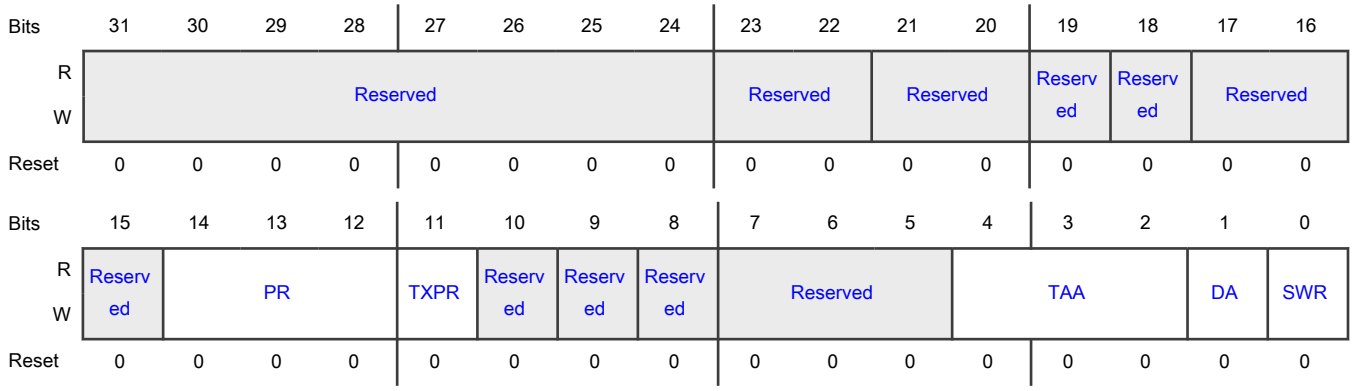
Offset

Register	Offset
DMA_MODE	1000h

Function

The Bus Mode register establishes the bus operating modes for the DMA.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-22 —	Reserved
21-20 —	Reserved
19 —	Reserved
18 —	Reserved
17-16 —	Reserved
15 —	Reserved
14-12 PR	Priority Ratio These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set. 000b - The priority ratio is 1:1 001b - The priority ratio is 2:1 010b - The priority ratio is 3:1 011b - The priority ratio is 4:1 100b - The priority ratio is 5:1 101b - The priority ratio is 6:1 110b - The priority ratio is 7:1 111b - The priority ratio is 8:1
11 TXPR	Transmit Priority When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus or Descriptor reads from DCACHE memory when DWC_EQOS_DCEXT is enabled 0b - Transmit Priority is disabled 1b - Transmit Priority is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 —	Reserved
9 —	Reserved
8 —	Reserved
7-5 —	Reserved
4-2 TAA	Transmit Arbitration Algorithm This field is used to select the arbitration algorithm for the Transmit side when multiple Tx DMAs are selected. 000b - Fixed priority. (Channel 0 has the lowest priority and the last channel has the highest priority) 001b - Weighted Strict Priority (WSP) 010b - Weighted Round-Robin (WRR) 011b - Reserved (for 3'b011 to 3'b111)
1 DA	DMA Tx or Rx Arbitration Scheme This bit specifies the arbitration scheme between the Transmit and Receive paths of all channels: 0: Weighted Round-Robin with Rx:Tx or Tx:Rx The priority between the paths is according to the priority specified in Bits[14:12] and the priority weight is specified in the TXPR bit. 1: Fixed Priority The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path. 0b - Weighted Round-Robin with Rx:Tx or Tx:Rx 1b - Fixed Priority
0 SWR	Software Reset When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all enet clock domains. Before reprogramming any enet register, a value of zero should be read in this bit. This bit must be read at least 4 CSR clock cycles after it is written to 1. Note: The reset operation is complete only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present

Table continues on the next page...

Table continued from the previous page...

Field	Function
	for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. 0b - Software Reset is disabled 1b - Software Reset is enabled

70.5.1.82 DMA System Bus Mode (DMA_SYSBUS_MODE)

Offset

Register	Offset
DMA_SYSBUS_MODE	1004h

Function

The System Bus mode register controls the behavior of the AHB or AXI master. It mainly controls burst splitting and number of outstanding requests.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	Reserved	Reserved				Reserved		Reserved						Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RB	MB	Reserved	AAL	Reserved	Reserved	Reserved		Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	FB
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29-26	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
25-24 —	Reserved
23-18 —	Reserved
17-16 —	Reserved
15 RB	Rebuild INCRx Burst When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds the pending beats of an EBT with an unspecified (INCR) burst. 0b - Rebuild INCRx Burst is disabled 1b - Rebuild INCRx Burst is enabled
14 MB	Mixed Burst When this bit is high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE). 0b - Mixed Burst is disabled 1b - Mixed Burst is enabled
13 —	Reserved
12 AAL	Address-Aligned Beats When this bit is set to 1, the EQOS-AXI or EQOS-AHB master performs address-aligned burst transfers on Read and Write channels. When this bit is set to 0, the EQOS-AXI or EQOS-AHB master performs burst transfers on Read and Write channels without aligning to address boundaries. 0b - Address-Aligned Beats is disabled 1b - Address-Aligned Beats is enabled
11 —	Reserved
10 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 FB	Fixed Burst Length 1: AHB master initiates burst transfers of specified length (INCRx or SINGLE). 0: AHB master initiates transfers of unspecified length (INCR) or SINGLE transfers. 0b - Fixed Burst Length is disabled 1b - Fixed Burst Length is enabled

70.5.1.83 DMA Interrupt Status (DMA_INTERRUPT_STATUS)

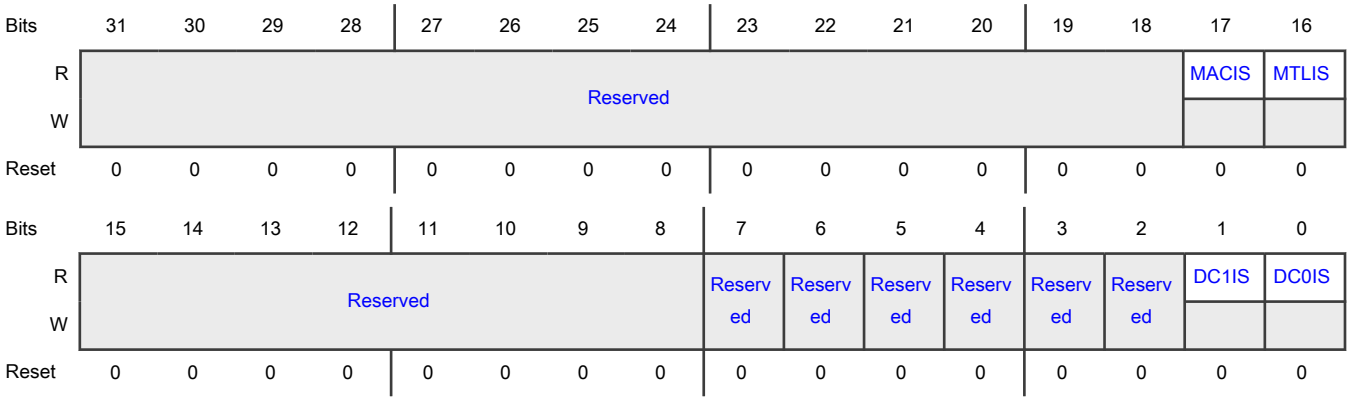
Offset

Register	Offset
DMA_INTERRUPT_STATUS	1008h

Function

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

Diagram



Fields

Field	Function
31-18 —	Reserved
17 MACIS	MAC Interrupt Status This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source. 0b - MAC Interrupt Status not detected 1b - MAC Interrupt Status detected
16 MTLIS	MTL Interrupt Status This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source. 0b - MTL Interrupt Status not detected 1b - MTL Interrupt Status detected
15-8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved
4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3 —	Reserved
2 —	Reserved
1 DC1IS	DMA Channel 1 Interrupt Status This bit indicates an interrupt event in DMA Channel 1. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 1 to get the exact cause of the interrupt and clear its source. 0b - DMA Channel 1 Interrupt Status not detected 1b - DMA Channel 1 Interrupt Status detected
0 DC0IS	DMA Channel 0 Interrupt Status This bit indicates an interrupt event in DMA Channel 0. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 0 to get the exact cause of the interrupt and clear its source. 0b - DMA Channel 0 Interrupt Status not detected 1b - DMA Channel 0 Interrupt Status detected

70.5.1.84 DMA Debug Status 0 (DMA_DEBUG_STATUS0)

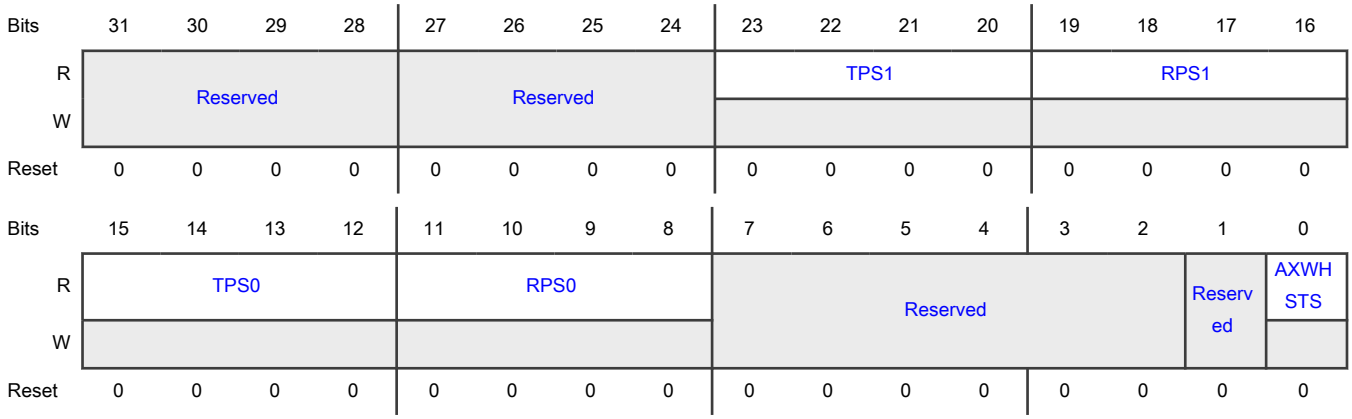
Offset

Register	Offset
DMA_DEBUG_STATUS0	100Ch

Function

The Debug Status 0 register gives the Receive and Transmit process status for DMA Channel 0-Channel 2 for debugging purpose.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 —	Reserved
23-20 TPS1	DMA Channel 1 Transmit Process State This field indicates the Tx DMA FSM state for Channel 1. The MSB of this field always returns 0. This field does not generate an interrupt. 0000b - Stopped (Reset or Stop Transmit Command issued) 0001b - Running (Fetching Tx Transfer Descriptor) 0010b - Running (Waiting for status) 0011b - Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO)) 0100b - Timestamp write state 0101b - Reserved for future use 0110b - Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow) 0111b - Running (Closing Tx Descriptor)
19-16 RPS1	DMA Channel 1 Receive Process State This field indicates the Rx DMA FSM state for Channel 1. The MSB of this field always returns 0. This field does not generate an interrupt. 0000b - Stopped (Reset or Stop Receive Command issued) 0001b - Running (Fetching Rx Transfer Descriptor) 0010b - Reserved for future use

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0011b - Running (Waiting for Rx packet) 0100b - Suspended (Rx Descriptor Unavailable) 0101b - Running (Closing the Rx Descriptor) 0110b - Timestamp write state 0111b - Running (Transferring the received packet data from the Rx buffer to the system memory)
15-12 TPS0	DMA Channel 0 Transmit Process State This field indicates the Tx DMA FSM state for Channel 0. The MSB of this field always returns 0. This field does not generate an interrupt. 0000b - Stopped (Reset or Stop Transmit Command issued) 0001b - Running (Fetching Tx Transfer Descriptor) 0010b - Running (Waiting for status) 0011b - Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO)) 0100b - Timestamp write state 0101b - Reserved for future use 0110b - Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow) 0111b - Running (Closing Tx Descriptor)
11-8 RPS0	DMA Channel 0 Receive Process State This field indicates the Rx DMA FSM state for Channel 0. The MSB of this field always returns 0. This field does not generate an interrupt. 0000b - Stopped (Reset or Stop Receive Command issued) 0001b - Running (Fetching Rx Transfer Descriptor) 0010b - Reserved for future use 0011b - Running (Waiting for Rx packet) 0100b - Suspended (Rx Descriptor Unavailable) 0101b - Running (Closing the Rx Descriptor) 0110b - Timestamp write state 0111b - Running (Transferring the received packet data from the Rx buffer to the system memory)
7-2 —	Reserved
1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 AXWHSTS	AHB Master Status When high, this bit indicates that the AHB master FSMs are in the non-idle state. 0b - AXI Master Write Channel or AHB Master Status not detected 1b - AXI Master Write Channel or AHB Master Status detected

70.5.1.85 DMA Channel 0 Control (DMA_CH0_CONTROL)

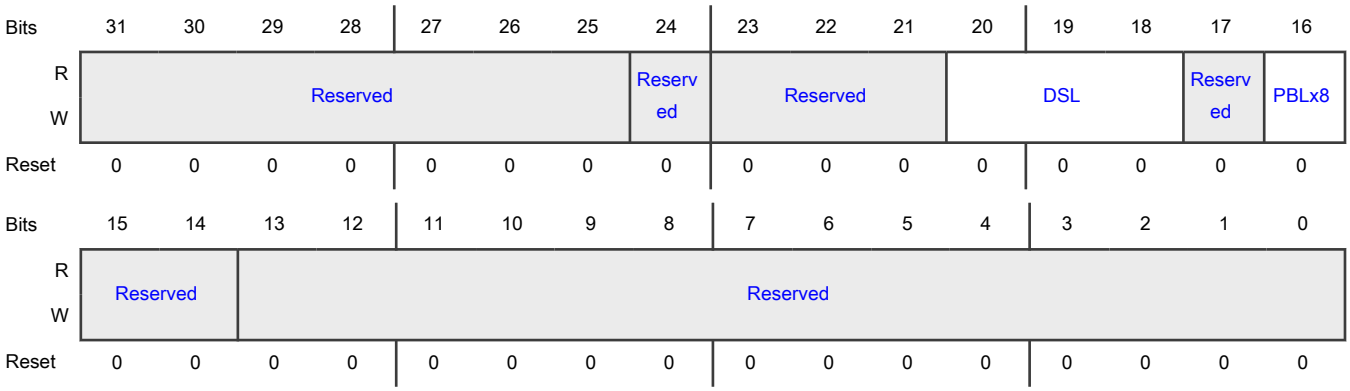
Offset

Register	Offset
DMA_CH0_CONTROL	1100h

Function

The DMA Channel 0 Control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

Diagram



Fields

Field	Function
31-25 —	Reserved
24 —	Reserved
23-21 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
20-18 DSL	Descriptor Skip Length This bit specifies the Word, Dword, or Lword number (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor. When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.
17 —	Reserved
16 PBLx8	8xPBL mode When this bit is set, the PBL value programmed in Bits[21:16] in DMA_CH(#i)_Tx_Control and Bits[21:16] in DMA_CH(#i)_Rx_Control is multiplied by eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value. 0b - 8xPBL mode is disabled 1b - 8xPBL mode is enabled
15-14 —	Reserved
13-0 —	Reserved

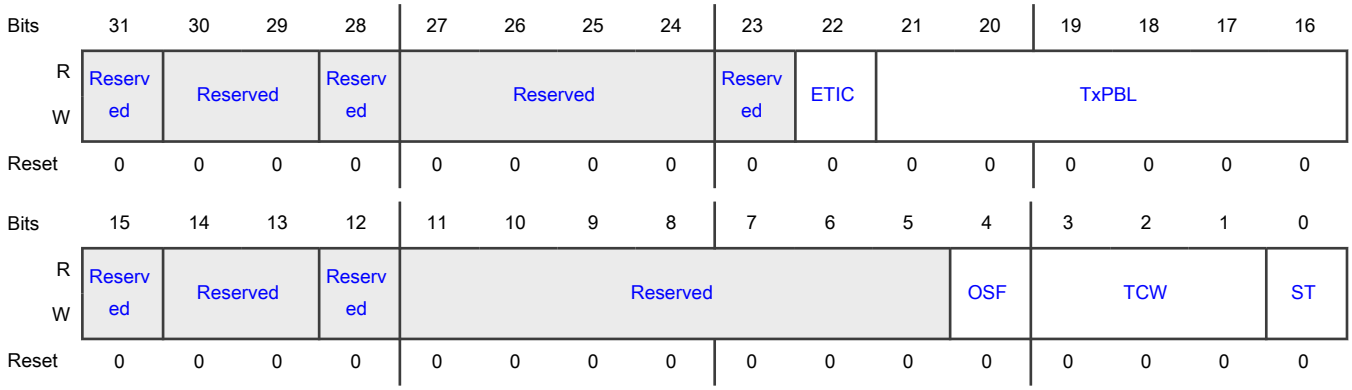
70.5.1.86 DMA Channel 0 Transmit Control (DMA_CH0_TX_CONTROL)

Offset

Register	Offset
DMA_CH0_TX_CONTROL	1104h

Function
The DMA Channel 0 Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

Diagram



Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 —	Reserved
27-24 —	Reserved
23 —	Reserved
22 ETIC	Early Transmit Interrupt Control When this bit is set, Early Transmit Interrupt (ETI) status is set after completion of transfer of data from buffers of a transmit descriptor in which IOC bit (TDES2[31]) is set. When this bit is reset, ETI is set only after a complete packet is transferred to the MTL TX FIFO memory. 0b - Early Transmit Interrupt is disabled 1b - Early Transmit Interrupt is enabled
21-16 TxPBL	Transmit Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in DMA_CH0_Control register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	2. Set the TxPBL. **NOTE** Note: The maximum value of TxPBL must be less than or equal to half the Tx Queue size (TQS field of MTL_TxQ(#i)_Operation_Mode register) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. For example, in 64-bit data width configurations the total locations in Tx Queue of size 512 bytes is 64, TxPBL and 8xPBL needs to be programmed to less than or equal to 32.
15 —	Reserved
14-13 —	Reserved
12 —	Reserved
11-5 —	Reserved
4 OSF	Operate on Second Packet When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained. 0b - Operate on Second Packet disabled 1b - Operate on Second Packet enabled
3-1 TCW	Transmit Channel Weight This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.
0 ST	Start or Stop Transmission Command When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions: The current position in the list This is the base address of the Transmit list set by the DMA_CH0_TxDesc_List_Address register. The position at which the transmission was previously stopped If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the DMA_CH0_Status register is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the DMA_CH0_TxDesc_List_Address register, the DMA behavior is unpredictable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program DMA_CH0_TxDesc_List_Address register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state. 0b - Stop Transmission Command 1b - Start Transmission Command

70.5.1.87 DMA Channel 0 Receive Control (DMA_CH0_RX_CONTROL)

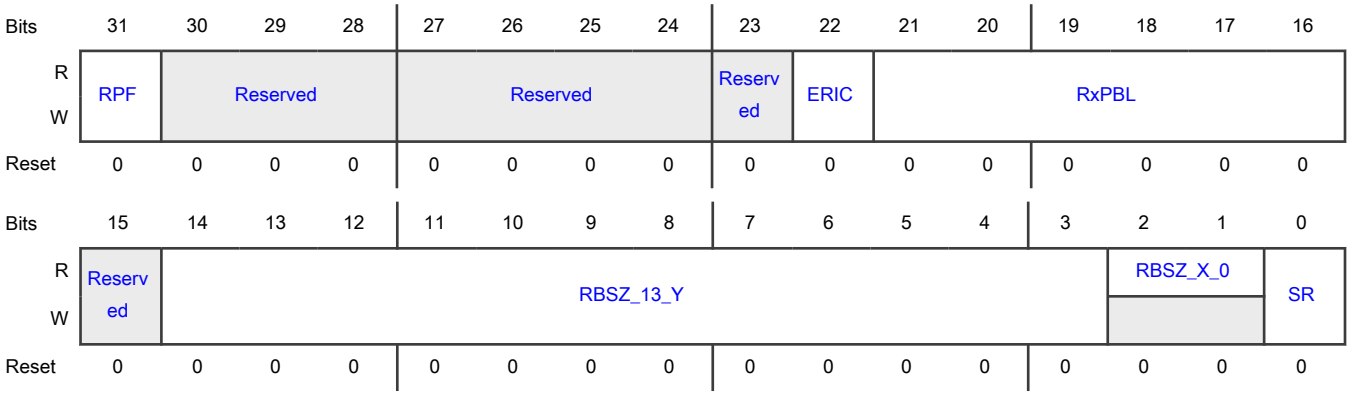
Offset

Register	Offset
DMA_CH0_RX_CONTROL	1108h

Function

The DMA Channel 0 Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

Diagram



Fields

Field	Function
31 RPF	Rx Packet Flush. 1: enet automatically flushes the packet from the Rx Queues destined to this DMA Rx Channel, when it is stopped. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are

Table continues on the next page...

Table continued from the previous page...

Field	Function
	received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing is done on the Read side of the Rx Queue. 0: enet does not flush the packet in the Rx Queue destined to this RxDMA Channel when it is in STOP state. This might cause head-of-line blocking in the corresponding RxQueue. Note: The stopping of packet flow from a Rx DMA Channel to the application by setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA channels. In Dynamic mapping mode, setting RPF bit in any DMA_CH(#i)_Rx_Control register might flush packets from unintended Rx Queues which are destined to the stopped Rx DMA Channel. 0b - Rx Packet Flush is disabled 1b - Rx Packet Flush is enabled
30-28 —	Reserved
27-24 —	Reserved
23 —	Reserved
22 ERIC	Early Receive Interrupt Control When this bit is set, Early Receive Interrupt (ERI) status is set after the completion of every burst transfer of data from the Rx DMA to the buffer. When this bit is reset, ERI is set only after a complete buffer is filled up by the RxDMA. 0b - Early Receive Interrupt is disabled 1b - Early Receive Interrupt is enabled
21-16 RxPBL	Receive Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. The PBL can be programmed with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in the DMA_CH0_Control register. 2. Set the RxPBL. Note: The maximum value of RxPBL must be less than or equal to half the Rx Queue size (RQS field of MTL_RxQn_Operation_Mode register) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the Rx DMA is transferring a block of data. For example, in 64-bit data width configurations the total locations in Rx Queue of size 512 bytes is 64, so RxPBL and 8xPBL needs to be programmed to less than or equal to 32.
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14-3 RBSZ_13_Y	Receive Buffer size High RBSZ[13:0] is split into two fields higher RBSZ_13_Y and lower RBSZ_X_0. The RBSZ[13:0] field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. The buffer size is applicable to payload buffers when split headers are enabled. Note: The buffer size must be a multiple of 4, 8, or 16 depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively). This is required even if the value of buffer address pointer is not aligned to data bus width. Hence the lower RBSZ_X_0 bits are read-only and the value is considered as all-zero. Thus the RBSZ_13_Y indicates the buffer size in terms of locations (with the width same as bus-width).
2-1 RBSZ_X_0	Receive Buffer size Low RBSZ[13:0] is split into two fields RBSZ_13_Y and RBSZ_X_0. The RBSZ_X_0 is the lower field whose width is based on data bus width of the configuration. This field is of width 2, 3, or 4 bits for 32-bit, 64-bit, or 128-bit data bus width respectively. This field is read-only (RO).
0 SR	Start or Stop Receive When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets. The DMA tries to acquire descriptor from either of the following positions: The current position in the list This is the address set by the DMA_CH0_RxDesc_List_Address register. The position at which the Rx process was previously stopped If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the DMA_CH0_Status register is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the DMA_CH0_RxDesc_List_Address register, the DMA behavior is unpredictable. When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state. 0b - Stop Receive 1b - Start Receive

70.5.1.88 Channel 0 Tx Descriptor List Address register (DMA_CH0_TXDESC_LIST_ADDRESS)

Offset

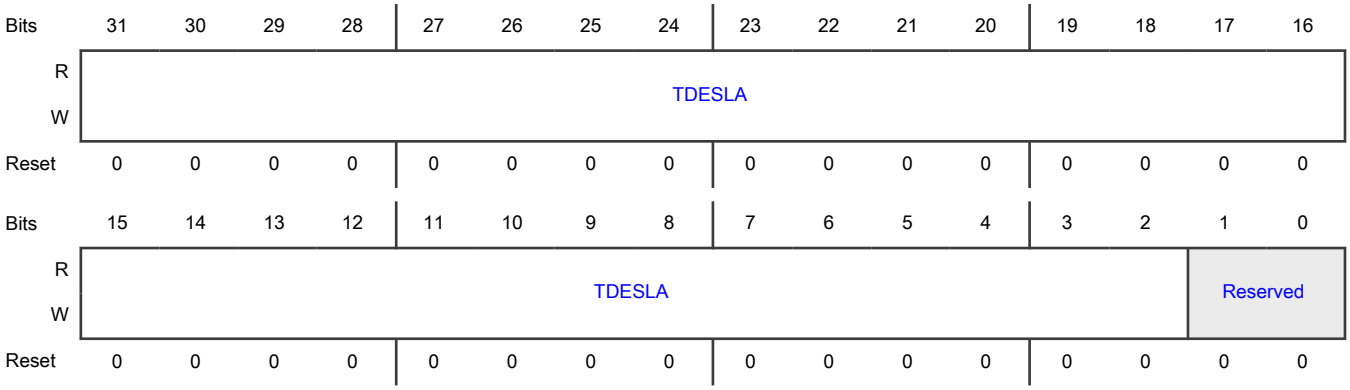
Register	Offset
DMA_CH0_TXDESC_LIST_ADDRESS	1114h

Function

The Channel 0 Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, the ST bit is set to zero in TX_CONTROL register. When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly-programmed descriptor base address. If this register is not changed when the ST bit is set to 0, the DMA takes the descriptor address where it was stopped earlier.

Diagram



Fields

Field	Function
31-2 TDESLA	Start of Transmit List This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO). The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

70.5.1.89 Channel 0 Rx Descriptor List Address register (DMA_CH0_RXDESC_LIST_ADDRESS)

Offset

Register	Offset
DMA_CH0_RXDESC_LIST_ADDRESS	111Ch

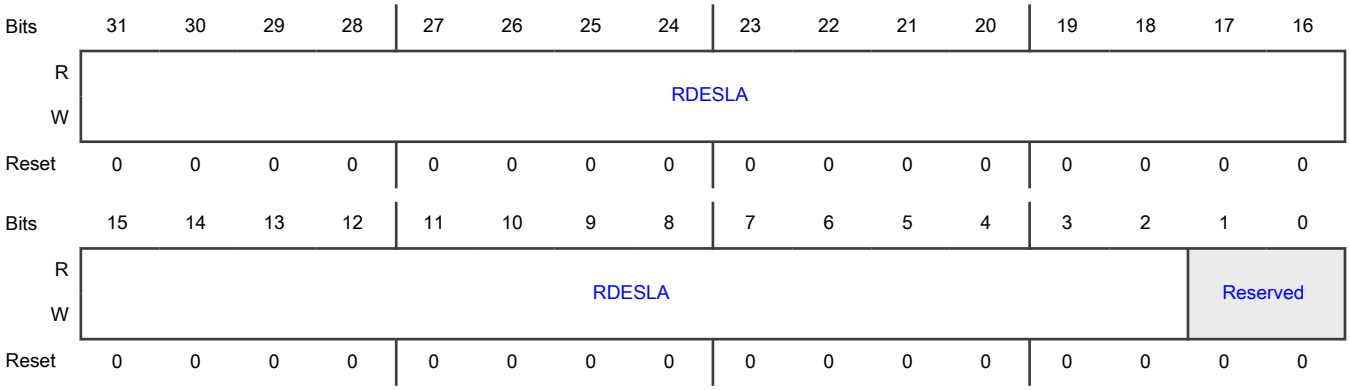
Function

The Channel 0 Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in RX_CONTROL register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

Diagram



Fields

Field	Function
31-2 RDESLA	Start of Receive List This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO). The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

70.5.1.90 Channel 0 Tx Descriptor Tail Pointer (DMA_CH0_TXDESC_TAIL_POINTER)

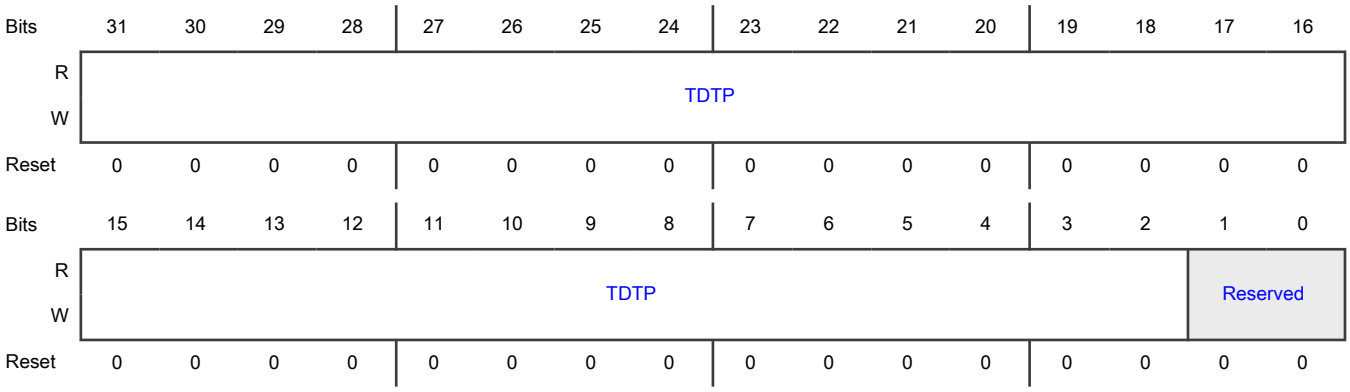
Offset

Register	Offset
DMA_CH0_TXDESC_TAIL_POINTER	1120h

Function

The Channel 0 Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

Diagram



Fields

Field	Function
31-2 TDTP	Transmit Descriptor Tail Pointer This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers. The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

70.5.1.91 Channel 0 Rx Descriptor Tail Pointer (DMA_CH0_RXDESC_TAIL_POINTER)

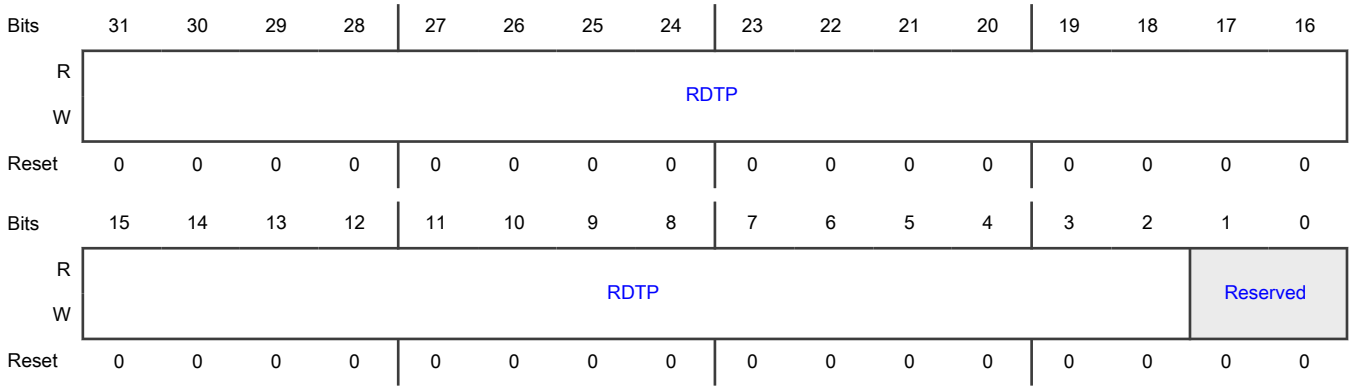
Offset

Register	Offset
DMA_CH0_RXDESC_TAIL_POINTER	1128h

Function

The Channel 0 Rx Descriptor Tail Pointer points to an offset from the base and indicates the location of the last valid descriptor.

Diagram



Fields

Field	Function
31-2 RDTP	Receive Descriptor Tail Pointer This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers. The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

70.5.1.92 Channel 0 Tx Descriptor Ring Length (DMA_CH0_TXDESC_RING_LENGTH)

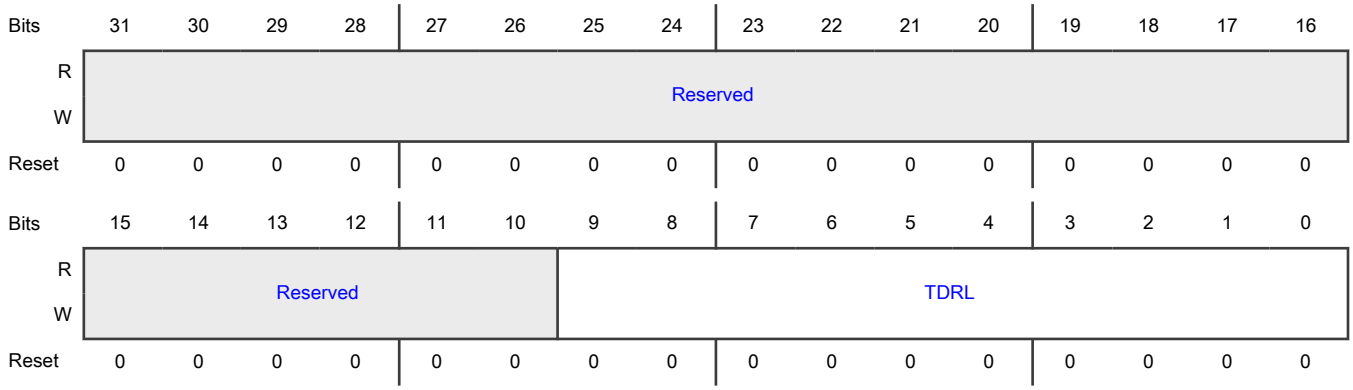
Offset

Register	Offset
DMA_CH0_TXDESC_RING_LENGTH	112Ch

Function

The Channel 0 Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 TDRL	Transmit Descriptor Ring Length This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. We recommends a minimum ring descriptor length of 4. For example, You can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

70.5.1.93 Channeli Receive Control (DMA_CH0_RX_CONTROL2)

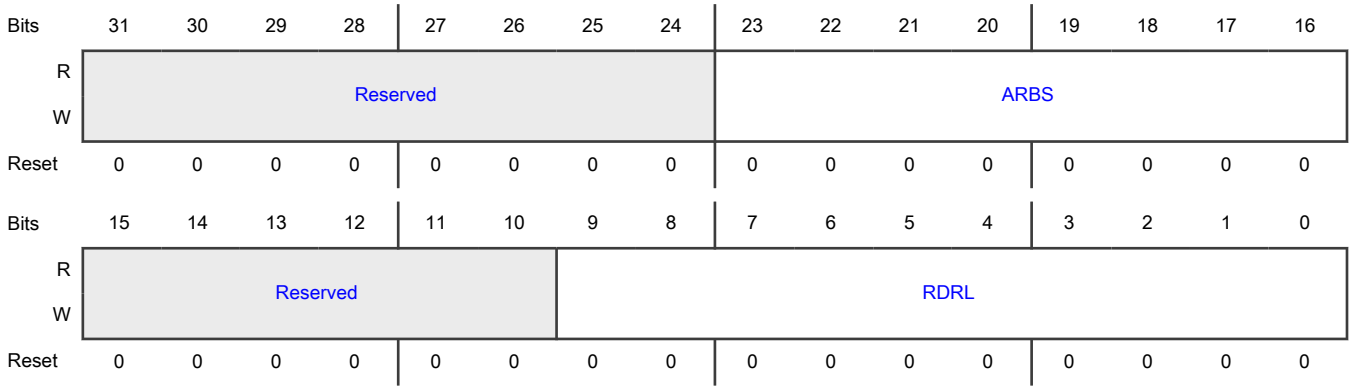
Offset

Register	Offset
DMA_CH0_RX_CONTROL2	1130h

Function

The Channeli Receive Control register controls the Rx features such as Rx Descriptor Ring Length and Alternate Rx Buffer Size.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 ARBS	Alternate Receive Buffer Size Indicates size in bytes for Buffer 1 when ARBS is programmed to a non-zero value (when split header feature is not enabled). When split header feature is enabled, ARBS indicates the buffer size for header data. The maximum alternate buffer is limited to 1020, 1016 or 1008-bytes depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively). When ARBS=0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ field of DMA_CH(#i)_Rx_Control. Width of ARBS field is 8, 7 or 6-bits depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively)
15-10 —	Reserved
9-0 RDRL	Receive Descriptor Ring Length This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. For example, You can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

70.5.1.94 Channeli Interrupt Enable (DMA_CH0_INTERRUPT_ENABLE)

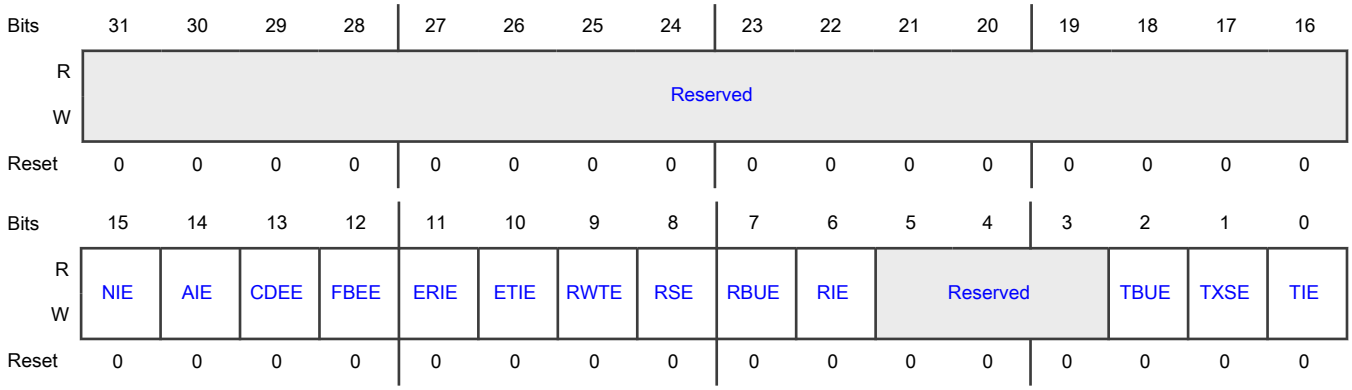
Offset

Register	Offset
DMA_CH0_INTERRUPT_ENABLE	1134h

Function

Enables the interrupts reported by Status register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 NIE	Normal Interrupt Summary Enable When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: Bit 0: Transmit Interrupt Bit 2: Transmit Buffer Unavailable Bit 6: Receive Interrupt Bit 11: Early Receive Interrupt When this bit is reset, the normal interrupt summary is disabled. 0b - Normal Interrupt Summary is disabled 1b - Normal Interrupt Summary is enabled
14 AIE	Abnormal Interrupt Summary Enable When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: Bit 1: Transmit Process Stopped Bit 7: Rx Buffer Unavailable Bit 8: Receive Process Stopped Bit 9: Receive Watchdog Timeout Bit 10: Early Transmit Interrupt Bit 12: Fatal Bus Error Bit 13: Context Descriptor Error When this bit is reset, the abnormal interrupt summary is disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Abnormal Interrupt Summary is disabled 1b - Abnormal Interrupt Summary is enabled
13 CDEE	Context Descriptor Error Enable When this bit is set along with the AIE bit, the Descriptor error interrupt is enabled. When this bit is reset, the Descriptor error interrupt is disabled. 0b - Context Descriptor Error is disabled 1b - Context Descriptor Error is enabled
12 FBEE	Fatal Bus Error Enable When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled. 0b - Fatal Bus Error is disabled 1b - Fatal Bus Error is enabled
11 ERIE	Early Receive Interrupt Enable When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled. 0b - Early Receive Interrupt is disabled 1b - Early Receive Interrupt is enabled
10 ETIE	Early Transmit Interrupt Enable When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled. 0b - Early Transmit Interrupt is disabled 1b - Early Transmit Interrupt is enabled
9 RWTE	Receive Watchdog Timeout Enable When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled. 0b - Receive Watchdog Timeout is disabled 1b - Receive Watchdog Timeout is enabled
8 RSE	Receive Stopped Enable When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled. 0b - Receive Stopped is disabled 1b - Receive Stopped is enabled
7	Receive Buffer Unavailable Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
RBUE	When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled. 0b - Receive Buffer Unavailable is disabled 1b - Receive Buffer Unavailable is enabled
6 RIE	Receive Interrupt Enable When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled. 0b - Receive Interrupt is disabled 1b - Receive Interrupt is enabled
5-3 —	Reserved
2 TBUE	Transmit Buffer Unavailable Enable When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled. 0b - Transmit Buffer Unavailable is disabled 1b - Transmit Buffer Unavailable is enabled
1 TXSE	Transmit Stopped Enable When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled. 0b - Transmit Stopped is disabled 1b - Transmit Stopped is enabled
0 TIE	Transmit Interrupt Enable When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled. 0b - Transmit Interrupt is disabled 1b - Transmit Interrupt is enabled

70.5.1.95 Channel 0 Receive Interrupt Watchdog Timer (DMA_CH0_RX_INTERRUPT_WATCHDOG_TIMER)

Offset

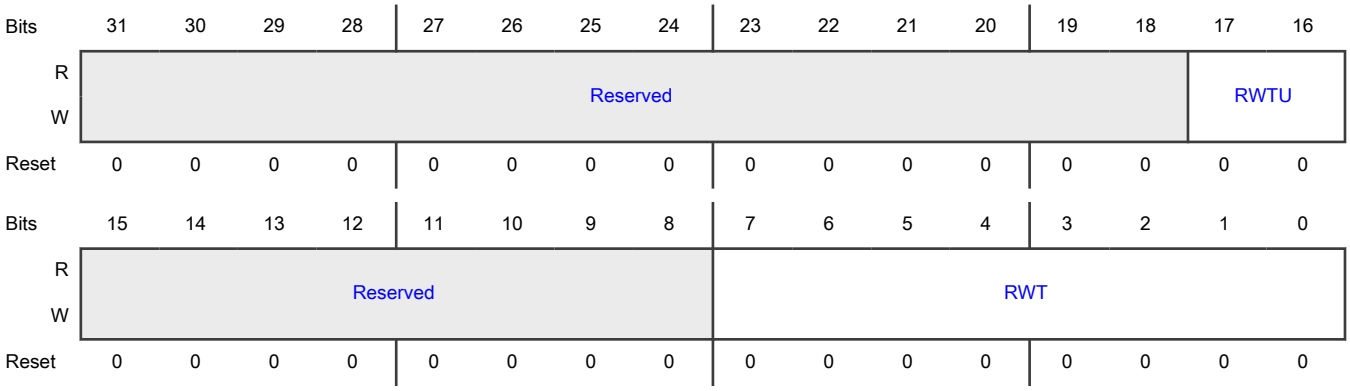
Register	Offset
DMA_CH0_RX_INTERRUPT_WATCHDOG_TIMER	1138h

Function

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA.

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the STATUS register.

Diagram



Fields

Field	Function
31-18 —	Reserved
17-16 RWTU	Receive Interrupt Watchdog Timer Count Units Receive Interrupt Watchdog Timer Count Units This fields indicates the number of system clock cycles corresponding to one unit in RWT field. 2'b00: 256 2'b01: 512 2'b10: 1024 2'b11: 2048 For example, when RWT=2 and RWTU=1, the watchdog timer is set for 2*512=1024 system clock cycles.
15-8 —	Reserved
7-0 RWT	Receive Interrupt Watchdog Timer Count This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set. The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the DMA_CH(#i)_Status register, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

70.5.1.96 Channel 0 Slot Function Control and Status (DMA_CH0_SLOT_FUNCTION_CONTROL_STATUS)

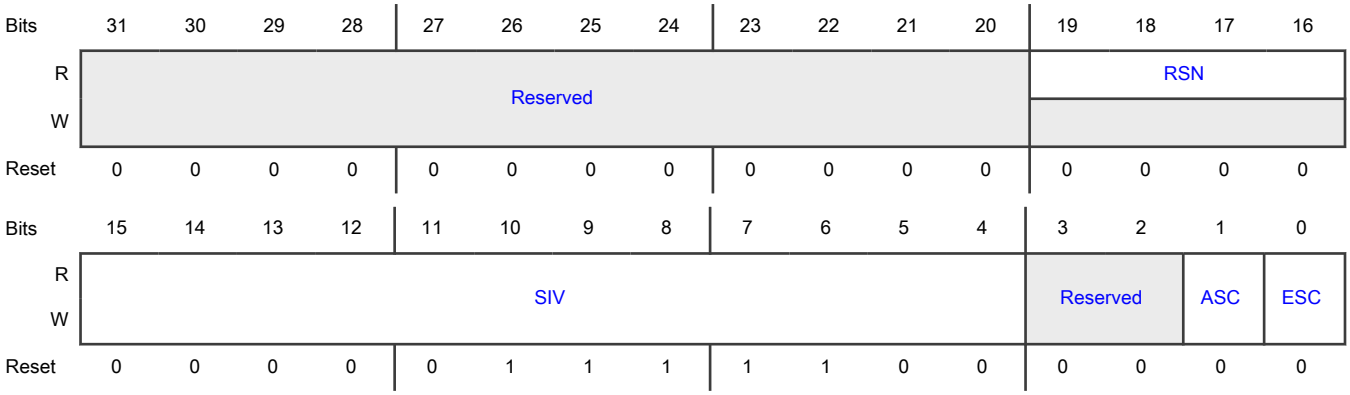
Offset

Register	Offset
DMA_CH0_SLOT_FUNCTION_CONTROL_STATUS	113Ch

Function

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 RSN	Reference Slot Number This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.
15-4 SIV	Slot Interval Value This field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 us while the maximum value 4095 specifies the slot interval of 4096 us. The default/reset value is 0x07C which corresponds to slot interval of 125 us
3-2 —	Reserved
1 ASC	Advance Slot Check When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is

Table continues on the next page...

Table continued from the previous page...

Field	Function
	equal to the reference slot number given in the RSN field or ahead of the reference slot number by up to two slots This bit is applicable only when the ESC bit is set. 0b - Advance Slot Check is disabled 1b - Advance Slot Check is enabled
0 ESC	Enable Slot Comparison When set, this bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is equal to the reference slot number or ahead of the reference slot number by one slot When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed. The UFO (UDP Fragmentation over IPv4)/TSO/USO should not be enabled along with TBS/AVB Slot number check. The UFO/TSO/USO involves multiple packets/segments/fragments transmission for single packet received from application and the slot number check are applicable for fetching of only first segment/fragment. As a result it might be difficult for software to specify slot number for subsequent packets. 0b - Slot Comparison is disabled 1b - Slot Comparison is enabled

70.5.1.97 Channel 0 Current Application Transmit Descriptor (DMA_CH0_CURRENT_APP_TXDESC)

Offset

Register	Offset
DMA_CH0_CURRENT_APP_TXDESC	1144h

Function

The Channel 0 Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CURTDESAPTR	Application Transmit Descriptor Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

70.5.1.98 Channel 0 Current Application Receive Descriptor (DMA_CH0_CURRENT_APP_RXDESC)

Offset

Register	Offset
DMA_CH0_CURRENT_APP_RXDESC	114Ch

Function

The Channel 0 Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CURRDESAPTR	Application Receive Descriptor Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

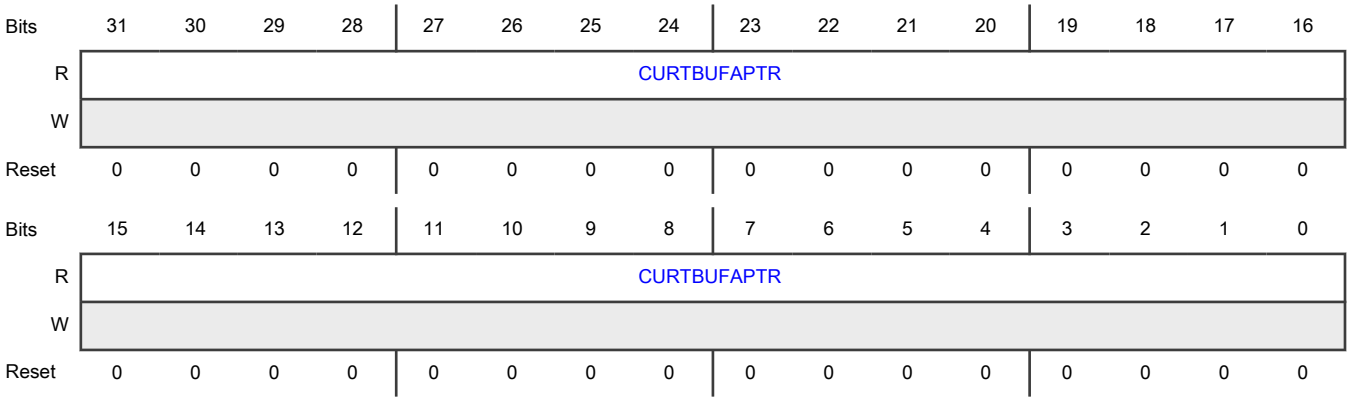
70.5.1.99 Channel 0 Current Application Transmit Buffer Address (DMA_CH0_CURRENT_APP_TXBUFFER)

Offset

Register	Offset
DMA_CH0_CURRENT_APP_TXBUFFER	1154h

Function
The Channel 0 Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

Diagram



Fields

Field	Function
31-0 CURTBUFAPTR	Application Transmit Buffer Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

70.5.1.100 Channel 0 Current Application Receive Buffer Address (DMA_CH0_CURRENT_APP_RXBUFFER)

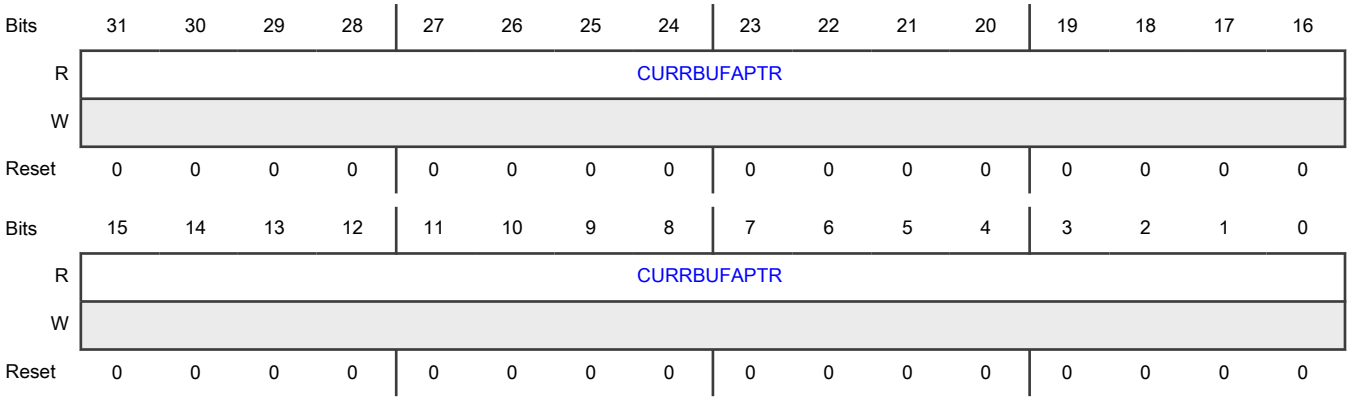
Offset

Register	Offset
DMA_CH0_CURRENT_APP_RXBUFFER	115Ch

Function

The Channel 0 Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

Diagram



Fields

Field	Function
31-0 CURRBUFAPTR	Application Receive Buffer Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

70.5.1.101 DMA Channel 0 Status (DMA_CH0_STATUS)

Offset

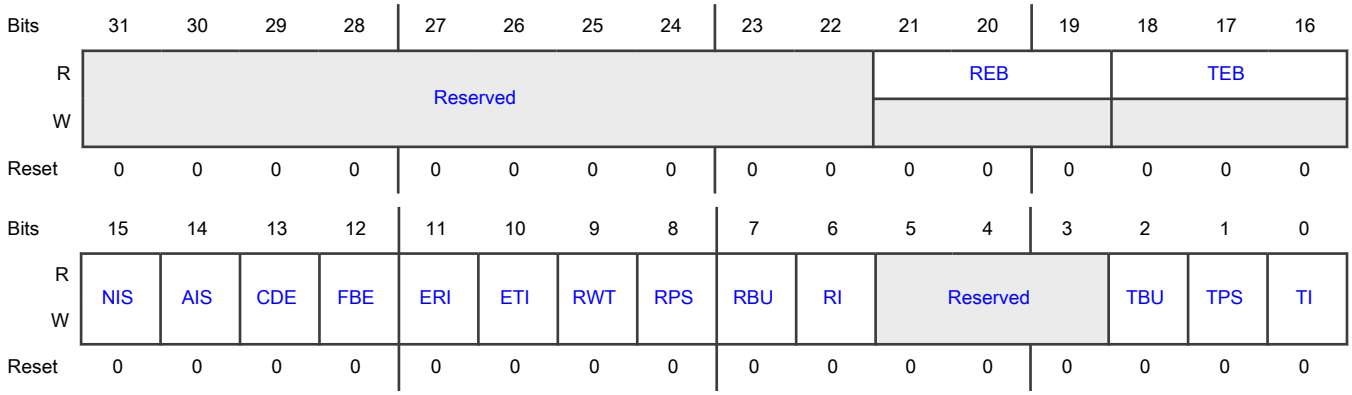
Register	Offset
DMA_CH0_STATUS	1160h

Function

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

Note: The number of STATUS register in the configuration is the higher of number of Rx DMA Channels and Tx DMA Channels.

Diagram



Fields

Field	Function
31-22 —	Reserved
21-19 REB	Rx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB or AXI interface. Bit 21 - 1'b1: Error during data transfer by Rx DMA - 1'b0: No Error during data transfer by Rx DMA Bit 20 - 1'b1: Error during descriptor access - 1'b0: Error during data buffer access Bit 19 - 1'b1: Error during read transfer - 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.
18-16 TEB	Tx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB or AXI interface. Bit 18 - 1'b1: Error during data transfer by Tx DMA - 1'b0: No Error during data transfer by Tx DMA Bit 17 - 1'b1: Error during descriptor access

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- 1'b0: Error during data buffer access Bit 16 - 1'b1: Error during read transfer - 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.
15 NIS	Normal Interrupt Summary Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: Bit 0: Transmit Interrupt Bit 2: Transmit Buffer Unavailable Bit 6: Receive Interrupt Bit 11: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in DMA_CH0_Interrupt_Enable register) affect the Normal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Normal Interrupt Summary status not detected 1b - Normal Interrupt Summary status detected
14 AIS	Abnormal Interrupt Summary Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the DMA_CH0_INTERRUPT_ENABLE register: Bit 1: Transmit Process Stopped Bit 7: Receive Buffer Unavailable Bit 8: Receive Process Stopped Bit 10: Early Transmit Interrupt Bit 12: Fatal Bus Error Bit 13: Context Descriptor Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Abnormal Interrupt Summary status not detected 1b - Abnormal Interrupt Summary status detected

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 CDE	Context Descriptor Error This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one's descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Context Descriptor Error status not detected 1b - Context Descriptor Error status detected
12 FBE	Fatal Bus Error This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Fatal Bus Error status not detected 1b - Fatal Bus Error status detected
11 ERI	Early Receive Interrupt This bit when set indicates that the RxDMA has completed the transfer of packet data to the memory. In configurations supporting ERIC, ERIC = 0: This bit is set only after the Rx DMA has completely filled up a receive buffer with packet data. ERIC = 1: This bit is set after every burst transfer of data from the Rx DMA to the buffer. The setting of RI bit automatically clears this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Early Receive Interrupt status not detected 1b - Early Receive Interrupt status detected
10 ETI	Early Transmit Interrupt This bit when set indicates that the TxDMA has completed the transfer of packet data to the MTL TXFIFO memory. In configurations supporting ERIC: ETIC = 0: This bit is set only after the Tx DMA has transferred a complete packet to MTL. ETIC = 1: This bit is set after completion of (partial) packet data transfer from buffers in the Transmit descriptor in which IOC = 1. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Early Transmit Interrupt status not detected 1b - Early Transmit Interrupt status detected
9	Receive Watchdog Timeout

Table continues on the next page...

Table continued from the previous page...

Field	Function
RWT	This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received. 0b - Receive Watchdog Timeout status not detected 1b - Receive Watchdog Timeout status detected
8 RPS	Receive Process Stopped This bit is asserted when the Rx process enters the Stopped state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Process Stopped status not detected 1b - Receive Process Stopped status detected
7 RBU	Receive Buffer Unavailable This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Buffer Unavailable status not detected 1b - Receive Buffer Unavailable status detected
6 RI	Receive Interrupt This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the Running state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Interrupt status not detected 1b - Receive Interrupt status detected
5-3 —	Reserved
2 TBU	Transmit Buffer Unavailable This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPS0 field of the DMA_Debug_Status0 register explains the Transmit Process state transitions. To resume processing the Transmit descriptors, the application should do the following: Change the ownership of the descriptor by setting Bit 31 of TDES3. Issue a Transmit Poll Demand command.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Buffer Unavailable status not detected 1b - Transmit Buffer Unavailable status detected
1 TPS	Transmit Process Stopped This bit is set when the transmission is stopped. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Process Stopped status not detected 1b - Transmit Process Stopped status detected
0 TI	Transmit Interrupt This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Interrupt status not detected 1b - Transmit Interrupt status detected

70.5.1.102 Channel 0 Missed Frame Counter (DMA_CH0_MISS_FRAME_CNT)

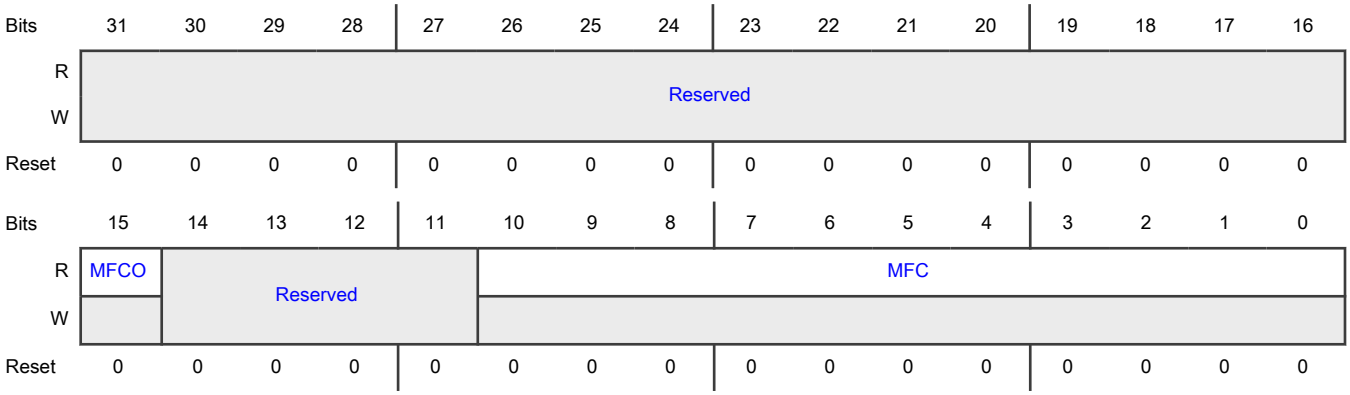
Offset

Register	Offset
DMA_CH0_MISS_FRAME_CNT	1164h

Function

This register has the number of packet counter that got dropped by the DMA either due to Bus Error or due to programming RPF field in RX_CONTROL register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 MFCO	Overflow status of the MFC Counter When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Miss Frame Counter overflow not occurred 1b - Miss Frame Counter overflow occurred
14-11 —	Reserved
10-0 MFC	Dropped Packet Counters This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in DMA_CH${i}_Rx_Control register. The counter gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.

70.5.1.103 Channel 0 Receive ERI Counter (DMA_CH0_RX_ERI_CNT)

Offset

Register	Offset
DMA_CH0_RX_ERI_CNT	116Ch

Function

The DMA_CH0_RX_ERI_CNT register provides the count of the number of times ERI was asserted.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				ECNT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11-0 ECNT	ERI Counter When ERIC bit of RX_CONTROL register is set, this counter increments for burst transfer completed by the Rx DMA from the start of packet transfer. This counter resets at the start of new packet.

70.5.1.104 DMA Channel 1 Control (DMA_CH1_CONTROL)

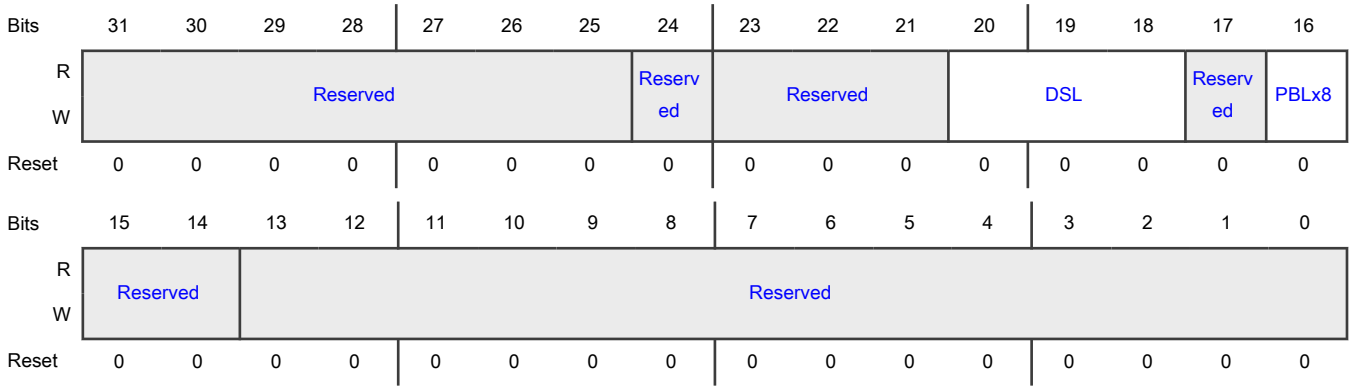
Offset

Register	Offset
DMA_CH1_CONTROL	1180h

Function

The DMA Channel 1 Control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

Diagram



Fields

Field	Function
31-25 —	Reserved
24 —	Reserved
23-21 —	Reserved
20-18 DSL	Descriptor Skip Length Descriptor Skip Length This bit specifies the Word, Dword, or Lword number (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor. When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.
17 —	Reserved
16 PBLx8	8xPBL mode 8xPBL mode When this bit is set, the PBL value programmed in Bits[21:16] in TX_CONTROL and Bits[21:16] in DMA_CH1_RX_CONTROL is multiplied by eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value. 0b - 8xPBL mode is disabled 1b - 8xPBL mode is enabled
15-14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-0 —	Reserved

70.5.1.105 DMA Channel 1 Transmit Control (DMA_CH1_TX_CONTROL)

Offset

Register	Offset
DMA_CH1_TX_CONTROL	1184h

Function

The DMA Channel 1 Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	Reserved		Reserved	Reserved				Reserved	ETIC	TxPBL					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	Reserved		Reserved	Reserved							OSF	TCW			ST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 —	Reserved
27-24	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
23 —	Reserved
22 ETIC	Early Transmit Interrupt Control When this bit is set, Early Transmit Interrupt (ETI) status is set after completion of transfer of data from buffers of a transmit descriptor in which IOC bit (TDES2[31]) is set. When this bit is reset, ETI is set only after a complete packet is transferred to the MTL TX FIFO memory. 0b - Early Transmit Interrupt is disabled 1b - Early Transmit Interrupt is enabled
21-16 TxPBL	Transmit Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: • Set the 8xPBL mode in DMA_CH0_Control register. • Set the TxPBL. **NOTE** The maximum value of TxPBL must be less than or equal to half the Tx Queue size (TQS field of MTL_TxQn Operation_Mode register) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. For example, in 64-bit data width configurations the total locations in Tx Queue of size 512 bytes is 64, TxPBL and 8xPBL needs to be programmed to less than or equal to 32.
15 —	Reserved
14-13 —	Reserved
12 —	Reserved
11-5 —	Reserved
4	Operate on Second Packet

Table continues on the next page...

Table continued from the previous page...

Field	Function
OSF	When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained. 0b - Operate on Second Packet disabled 1b - Operate on Second Packet enabled
3-1 TCW	Transmit Channel Weight This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.
0 ST	Start or Stop Transmission Command When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions: The current position in the list This is the base address of the Transmit list set by the DMA_CH0_TXDESC_LIST_ADDRESS register. The position at which the transmission was previously stopped If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the DMA_CH0_STATUS register is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the DMA_CH0_TXDESC_LIST_ADDRESS register, the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program DMA_CH0_TXDESC_LIST_ADDRESS register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state. 0b - Stop Transmission Command 1b - Start Transmission Command

70.5.1.106 DMA Channel 1 Receive Control (DMA_CH1_RX_CONTROL)

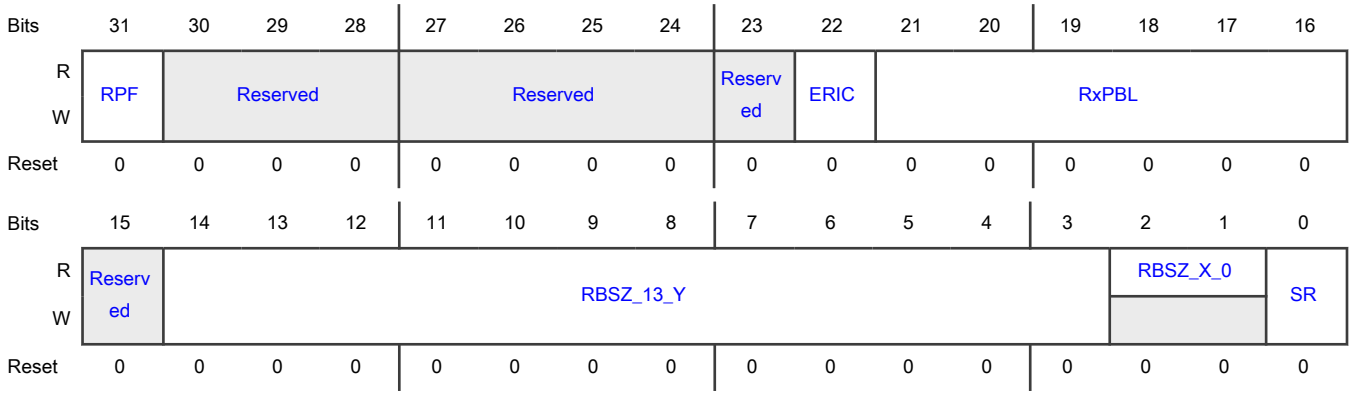
Offset

Register	Offset
DMA_CH1_RX_CONTROL	1188h

Function

The DMA Channel 1 Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

Diagram



Fields

Field	Function
31 RPF	Rx Packet Flush. When this bit is set to 1, then module automatically flushes the packet from the Rx Queues destined to this DMA Rx Channel, when it is stopped. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing is done on the Read side of the Rx Queue. When this bit is set to 0, module does not flush the packet in the Rx Queue destined to this RxDMA Channel when it is in STOP state. This might cause head-of-line blocking in the corresponding RxQueue. **NOTE** The stopping of packet flow from a Rx DMA Channel to the application by setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA channels. In Dynamic mapping mode, setting RPF bit in any DMA_CH(#i)_Rx_Control register might flush packets from unintended Rx Queues which are destined to the stopped Rx DMA Channel. 0b - Rx Packet Flush is disabled 1b - Rx Packet Flush is enabled
30-28 —	Reserved
27-24 —	Reserved
23 —	Reserved
22 ERIC	Early Receive Interrupt Control When this bit is set, Early Receive Interrupt (ERI) status is set after the completion of every burst transfer of data from the Rx DMA to the buffer. When this bit is reset, ERI is set only after a complete buffer is filled up by the RxDMA.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Early Receive Interrupt is disabled 1b - Early Receive Interrupt is enabled
21-16 RxPBL	Receive Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in the DMA_CH0_CONTROL register. 2. Set the RxPBL. Note: The maximum value of RxPBL must be less than or equal to half the Rx Queue size (RQS field of MTL_RxQn_OPERATION_MODE register) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the Rx DMA is transferring a block of data. For example, in 64-bit data width configurations the total locations in Rx Queue of size 512 bytes is 64, so RxPBL and 8xPBL needs to be programmed to less than or equal to 32.
15 —	Reserved
14-3 RBSZ_13_Y	Receive Buffer size High RBSZ[13:0] is split into two fields higher RBSZ_13_Y and lower RBSZ_X_0. The RBSZ[13:0] field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. The buffer size is applicable to payload buffers when split headers are enabled. Note: The buffer size must be a multiple of 4, 8, or 16 depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively). This is required even if the value of buffer address pointer is not aligned to data bus width. Hence the lower RBSZ_X_0 bits are read-only and the value is considered as all-zero. Thus the RBSZ_13_Y indicates the buffer size in terms of locations (with the width same as bus-width).
2-1 RBSZ_X_0	Receive Buffer size Low RBSZ[13:0] is split into two fields RBSZ_13_Y and RBSZ_X_0. The RBSZ_X_0 is the lower field whose width is based on data bus width of the configuration. This field is of width 2, 3, or 4 bits for 32-bit, 64-bit, or 128-bit data bus width respectively. This field is read-only (RO).
0 SR	Start or Stop Receive When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets. The DMA tries to acquire descriptor from either of the following positions: The current position in the list This is the address set by the DMA_CH1_RXDESC_LIST_ADDRESS register. The position at which the Rx process was previously stopped

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the DMA_CH1_STATUS register is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the DMA_CH1_RXDESC_LIST_ADDRESS register, the DMA behavior is unpredictable. When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state. 0b - Stop Receive 1b - Start Receive

70.5.1.107 Channel 1 Tx Descriptor List Address (DMA_CH1_TXDESC_LIST_ADDRESS)

Offset

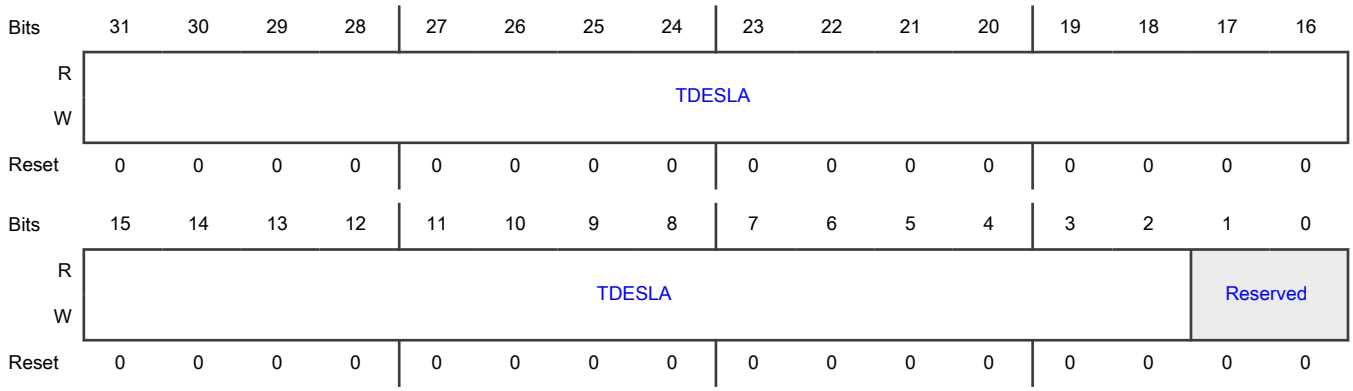
Register	Offset
DMA_CH1_TXDESC_LIST_ADDRESS	1194h

Function

The Channel 1 Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, the ST bit is set to zero in DMA_CH1_TX_CONTROL register. When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly-programmed descriptor base address. If this register is not changed when the ST bit is set to 0, the DMA takes the descriptor address where it was stopped earlier.

Diagram



Fields

Field	Function
31-2 TDESLA	Start of Transmit List This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO). The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

70.5.1.108 Channel 1 Rx Descriptor List Address (DMA_CH1_RXDESC_LIST_ADDRESS)

Offset

Register	Offset
DMA_CH1_RXDESC_LIST_ADDRESS	119Ch

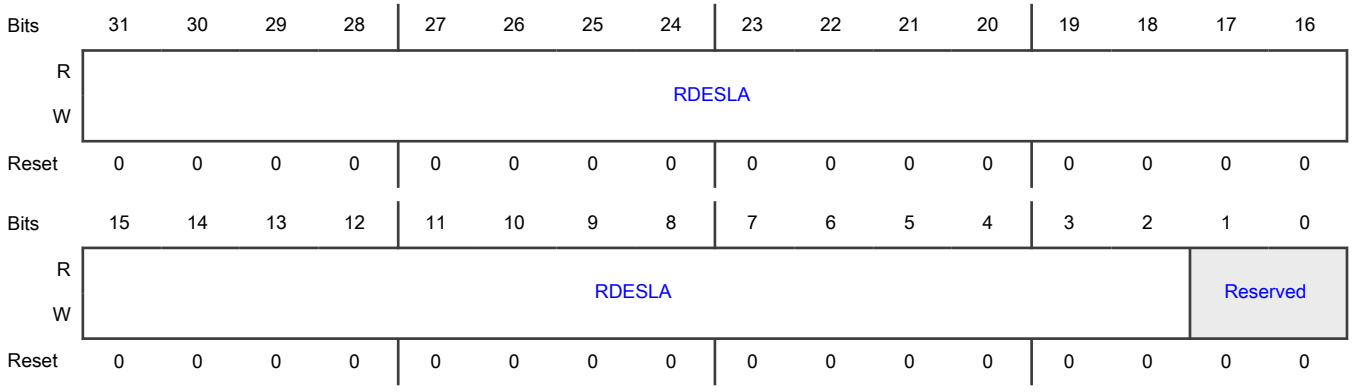
Function

The Channel 1 Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in DMA_CH1_RX_CONTROL register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

Diagram



Fields

Field	Function
31-2 RDESLA	Start of Receive List This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO). The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

70.5.1.109 Channel 1 Tx Descriptor Tail Pointer (DMA_CH1_TXDESC_TAIL_POINTER)

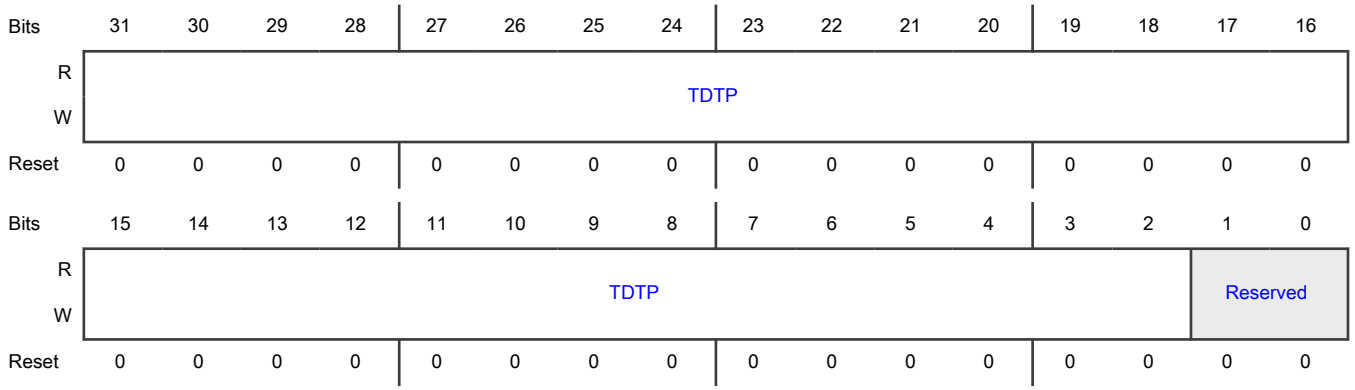
Offset

Register	Offset
DMA_CH1_TXDESC_TAIL_POINTER	11A0h

Function

The Channel 1 Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

Diagram



Fields

Field	Function
31-2 TDTP	Transmit Descriptor Tail Pointer This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers. The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

70.5.1.110 Channel 1 Rx Descriptor Tail Pointer (DMA_CH1_RXDESC_TAIL_POINTER)

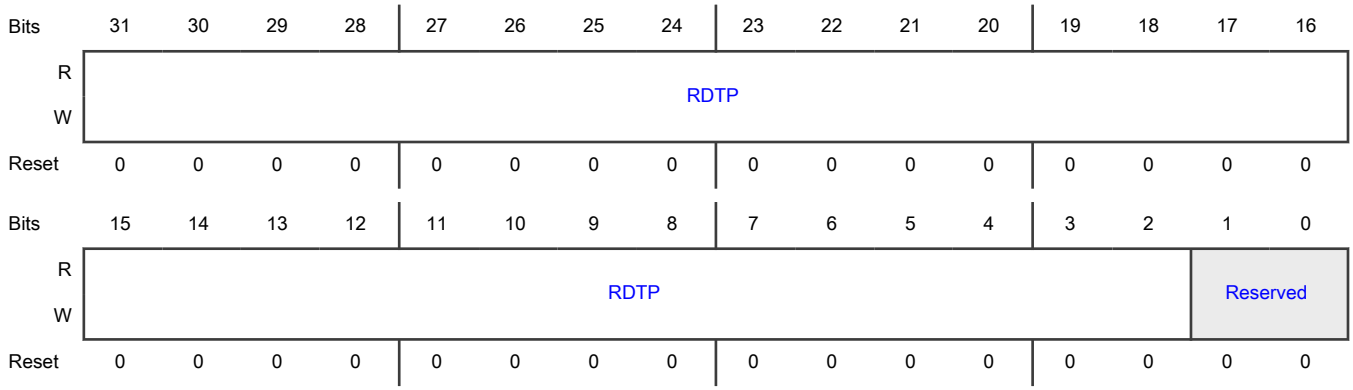
Offset

Register	Offset
DMA_CH1_RXDESC_TAIL_POINTER	11A8h

Function

The Channel 1 Rx Descriptor Tail Pointer points to an offset from the base and indicates the location of the last valid descriptor.

Diagram



Fields

Field	Function
31-2 RDTP	Receive Descriptor Tail Pointer This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers. The width of this field depends on the configuration: 31:2 for 32-bit configuration 31:3 for 64-bit configuration 31:4 for 128-bit configuration
1-0 —	Reserved

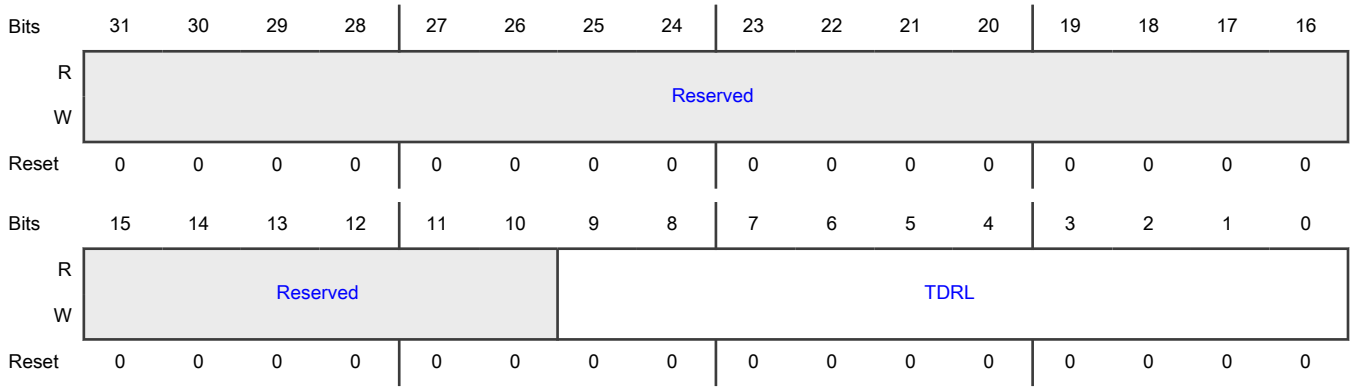
70.5.1.111 Channel 1 Tx Descriptor Ring Length (DMA_CH1_TXDESC_RING_LENGTH)

Offset

Register	Offset
DMA_CH1_TXDESC_RING_LENGTH	11ACh

Function
The Channel 1 Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

Diagram



Fields

Field	Function
31-10	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
9-0 TDRL	Transmit Descriptor Ring Length This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. We recommends a minimum ring descriptor length of 4. For example, You can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

70.5.1.112 DMA Channel 1 Receive Control (DMA_CH1_RX_CONTROL2)

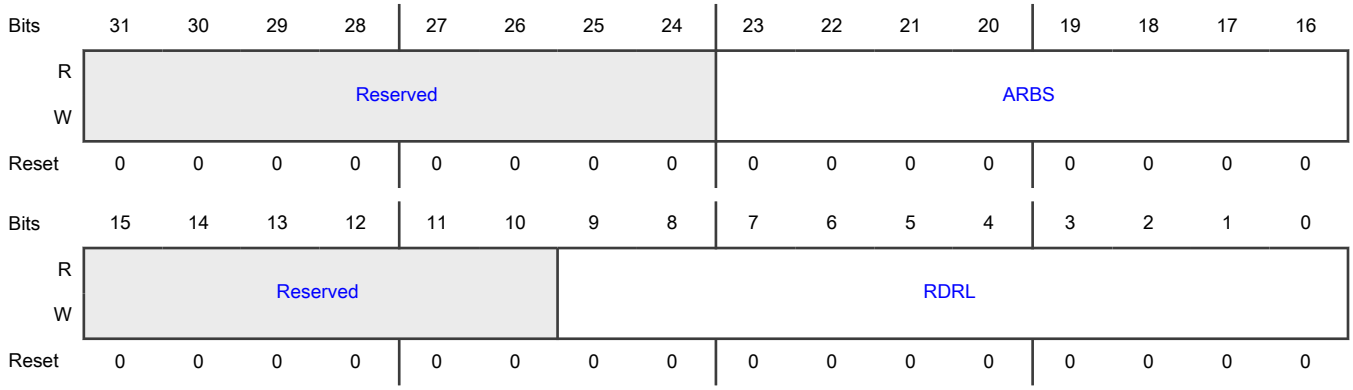
Offset

Register	Offset
DMA_CH1_RX_CONTROL2	11B0h

Function

The Channel 1 Receive Control register controls the Rx features such as Rx Descriptor Ring Length and Alternate Rx Buffer Size.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 ARBS	Alternate Receive Buffer Size

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates size in bytes for Buffer 1 when ARBS is programmed to a non-zero value (when split header feature is not enabled). When split header feature is enabled, ARBS indicates the buffer size for header data. The maximum alternate buffer is limited to 1020, 1016 or 1008-bytes depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively). When ARBS=0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ field of DMA_CH(#i)_Rx_Control. Width of ARBS field is 8, 7 or 6-bits depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively)
15-10 —	Reserved
9-0 RDRL	Receive Descriptor Ring Length This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. For example, You can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

70.5.1.113 Channel 1 Interrupt Enable (DMA_CH1_INTERRUPT_ENABLE)

Offset

Register	Offset
DMA_CH1_INTERRUPT_ENABLE	11B4h

Function

The Channel 1 Interrupt Enable register enables the interrupts reported by the Status register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Reserved			TBUE	TXSE	TIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15 NIE	Normal Interrupt Summary Enable When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: Bit 0: Transmit Interrupt Bit 2: Transmit Buffer Unavailable Bit 6: Receive Interrupt Bit 11: Early Receive Interrupt When this bit is reset, the normal interrupt summary is disabled. 0b - Normal Interrupt Summary is disabled 1b - Normal Interrupt Summary is enabled
14 AIE	Abnormal Interrupt Summary Enable When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: Bit 1: Transmit Process Stopped Bit 7: Rx Buffer Unavailable Bit 8: Receive Process Stopped Bit 9: Receive Watchdog Timeout Bit 10: Early Transmit Interrupt Bit 12: Fatal Bus Error Bit 13: Context Descriptor Error When this bit is reset, the abnormal interrupt summary is disabled. 0b - Abnormal Interrupt Summary is disabled 1b - Abnormal Interrupt Summary is enabled
13 CDEE	Context Descriptor Error Enable When this bit is set along with the AIE bit, the Descriptor error interrupt is enabled. When this bit is reset, the Descriptor error interrupt is disabled. 0b - Context Descriptor Error is disabled 1b - Context Descriptor Error is enabled
12 FBEE	Fatal Bus Error Enable When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Fatal Bus Error is disabled 1b - Fatal Bus Error is enabled
11 ERIE	Early Receive Interrupt Enable When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled. 0b - Early Receive Interrupt is disabled 1b - Early Receive Interrupt is enabled
10 ETIE	Early Transmit Interrupt Enable When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled. 0b - Early Transmit Interrupt is disabled 1b - Early Transmit Interrupt is enabled
9 RWTE	Receive Watchdog Timeout Enable When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled. 0b - Receive Watchdog Timeout is disabled 1b - Receive Watchdog Timeout is enabled
8 RSE	Receive Stopped Enable When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled. 0b - Receive Stopped is disabled 1b - Receive Stopped is enabled
7 RBUE	Receive Buffer Unavailable Enable When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled. 0b - Receive Buffer Unavailable is disabled 1b - Receive Buffer Unavailable is enabled
6 RIE	Receive Interrupt Enable When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled. 0b - Receive Interrupt is disabled 1b - Receive Interrupt is enabled
5-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2 TBUE	Transmit Buffer Unavailable Enable When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled. 0b - Transmit Buffer Unavailable is disabled 1b - Transmit Buffer Unavailable is enabled
1 TXSE	Transmit Stopped Enable When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled. 0b - Transmit Stopped is disabled 1b - Transmit Stopped is enabled
0 TIE	Transmit Interrupt Enable When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled. 0b - Transmit Interrupt is disabled 1b - Transmit Interrupt is enabled

70.5.1.114 Channel 1 Receive Interrupt Watchdog Timer (DMA_CH1_RX_INTERRUPT_WATCHDOG_TIMER)

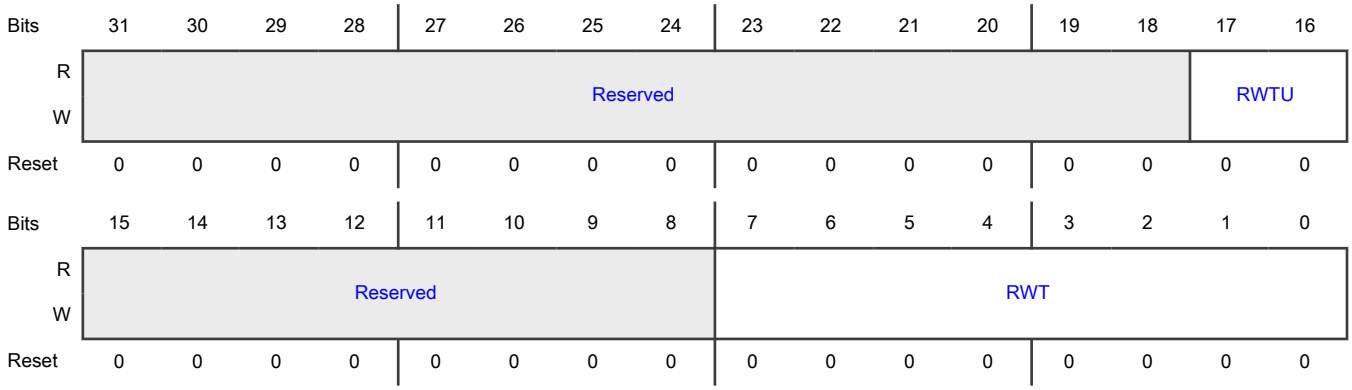
Offset

Register	Offset
DMA_CH1_RX_INTERRUPT_WATCHDOG_TIMER	11B8h

Function

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the STATUS register.

Diagram



Fields

Field	Function
31-18 —	Reserved
17-16 RWTU	Receive Interrupt Watchdog Timer Count Units This fields indicates the number of system clock cycles corresponding to one unit in RWT field. 2'b00: 256 2'b01: 512 2'b10: 1024 2'b11: 2048 For example, when RWT=2 and RWTU=1, the watchdog timer is set for 2*512=1024 system clock cycles.
15-8 —	Reserved
7-0 RWT	Receive Interrupt Watchdog Timer Count This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set. The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the DMA_CH(#i)_Status register, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

70.5.1.115 Channel 1 Slot Function Control and Status (DMA_CH1_SLOT_FUNCTION_CONTROL_STATUS)

Offset

Register	Offset
DMA_CH1_SLOT_FUNCTION_CONTROL_STATUS	11BCh

Function

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												RSN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SIV												Reserved		ASC	ESC
W																
Reset	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-16 RSN	Reference Slot Number This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.
15-4 SIV	Slot Interval Value This field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 us while the maximum value 4095 specifies the slot interval of 4096 us. The default/reset value is 0x07C which corresponds to slot interval of 125 us
3-2 —	Reserved
1 ASC	Advance Slot Check When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is

Table continues on the next page...

Table continued from the previous page...

Field	Function
	equal to the reference slot number given in the RSN field or ahead of the reference slot number by up to two slots This bit is applicable only when the ESC bit is set. 0b - Advance Slot Check is disabled 1b - Advance Slot Check is enabled
0 ESC	Enable Slot Comparison When set, this bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is equal to the reference slot number or ahead of the reference slot number by one slot When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed. **NOTE** The UFO (UDP Fragmentation over IPv4)/TSO/USO should not be enabled along with TBS/AVB Slot number check. The UFO/TSO/USO involves multiple packets/segments/ fragments transmission for single packet received from application and the slot number check are applicable for fetching of only first segment/fragment. As a result it might be difficult for software to specify slot number for subsequent packets. 0b - Slot Comparison is disabled 1b - Slot Comparison is enabled

70.5.1.116 Channel 1 Current Application Transmit Descriptor (DMA_CH1_CURRENT_APP_TXDESC)

Offset

Register	Offset
DMA_CH1_CURRENT_APP_TXDESC	11C4h

Function

The Channel 1 Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CURTDESAPTR	Application Transmit Descriptor Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

70.5.1.117 Channel 1 Current Application Receive Descriptor (DMA_CH1_CURRENT_APP_RXDESC)

Offset

Register	Offset
DMA_CH1_CURRENT_APP_RXDESC	11CCh

Function

The Channel 1 Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 CURRDESAPTR	Application Receive Descriptor Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

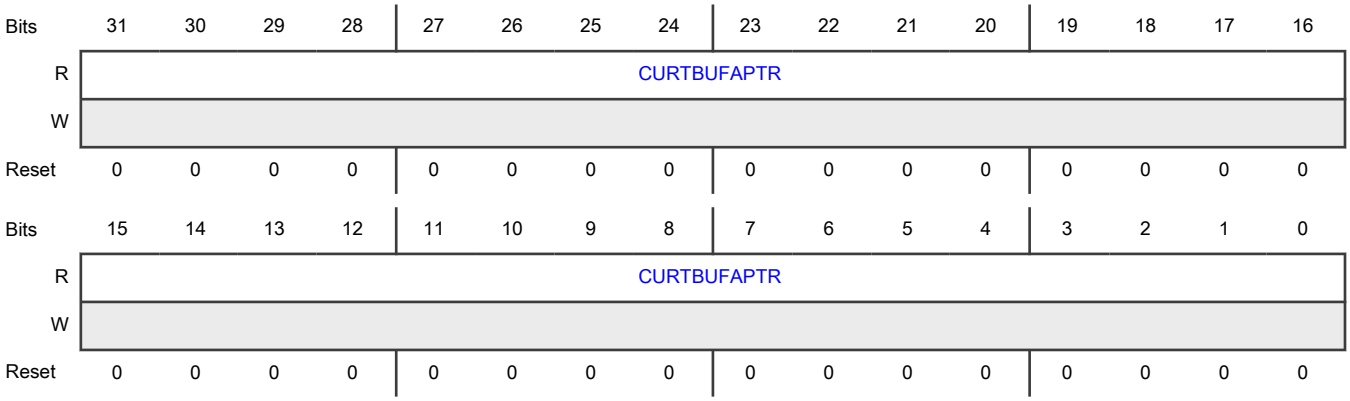
70.5.1.118 Channel 1 Current Application Transmit Buffer Address (DMA_CH1_CURRENT_APP_TXBUFFER)

Offset

Register	Offset
DMA_CH1_CURRENT_APP_TXBUFFER	11D4h

Function
The Channel 1 Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

Diagram



Fields

Field	Function
31-0 CURTBUFAPTR	Application Transmit Buffer Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

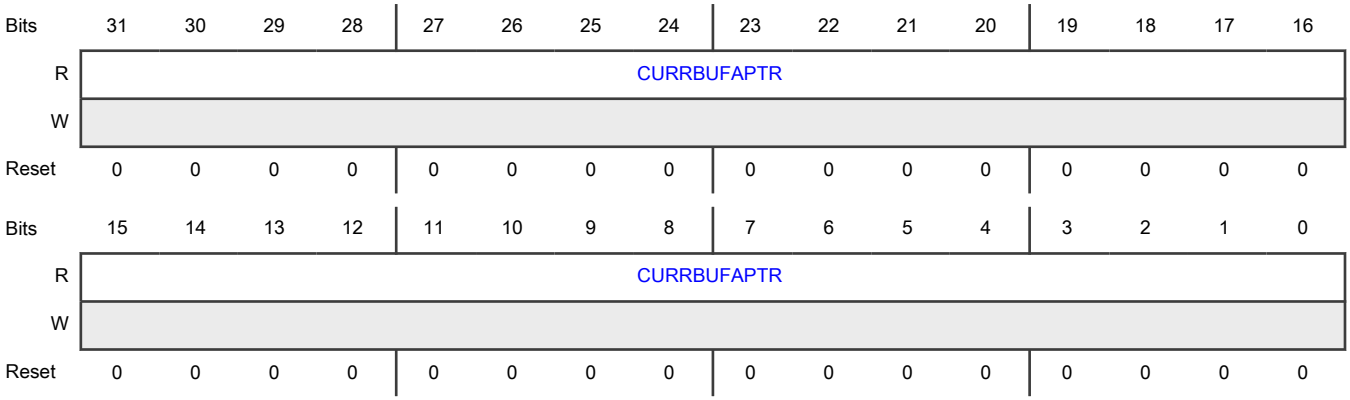
70.5.1.119 Channel 1 Current Application Receive Buffer Address (DMA_CH1_CURRENT_APP_RXBUFFER)

Offset

Register	Offset
DMA_CH1_CURRENT_APP_RXBUFFER	11DCh

Function
The Channel 1 Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

Diagram



Fields

Field	Function
31-0 CURRBUFAPTR	Application Receive Buffer Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

70.5.1.120 DMA Channel 1 Status (DMA_CH1_STATUS)

Offset

Register	Offset
DMA_CH1_STATUS	11E0h

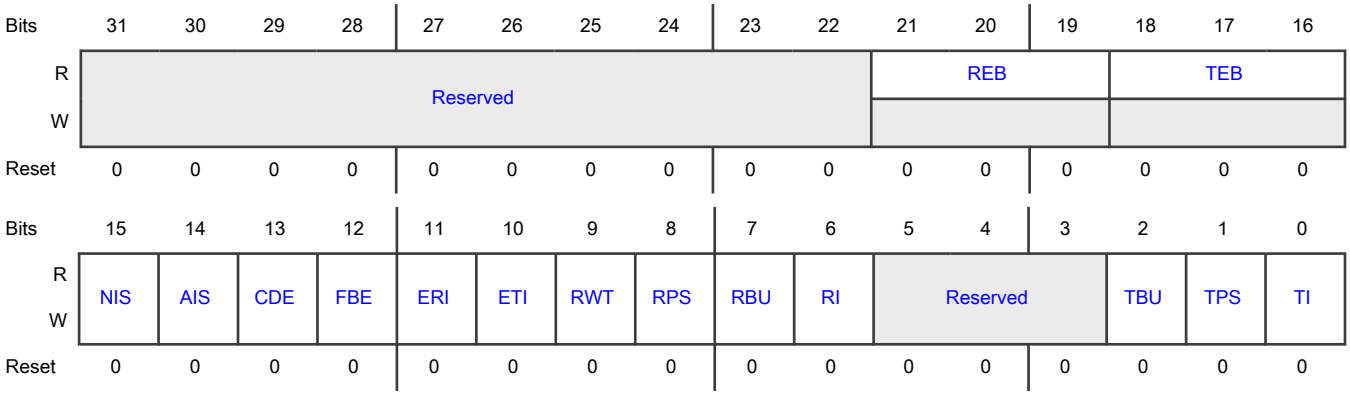
Function
The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

NOTE

The number of STATUS register in the configuration is the higher of number of Rx DMA Channels and Tx DMA Channels.

Diagram



Fields

Field	Function
31-22 —	Reserved
21-19 REB	Rx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB or AXI interface. Bit 21 1'b1: Error during data transfer by Rx DMA 1'b0: No Error during data transfer by Rx DMA Bit 20 1'b1: Error during descriptor access 1'b0: Error during data buffer access Bit 19 1'b1: Error during read transfer 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.
18-16 TEB	Tx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB or AXI interface. Bit 18 1'b1: Error during data transfer by Tx DMA

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1'b0: No Error during data transfer by Tx DMA Bit 17 1'b1: Error during descriptor access 1'b0: Error during data buffer access Bit 16 1'b1: Error during read transfer 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.
15 NIS	Normal Interrupt Summary Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: Bit 0: Transmit Interrupt Bit 2: Transmit Buffer Unavailable Bit 6: Receive Interrupt Bit 11: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in DMA_CH0_Interrupt_Enable register) affect the Normal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Normal Interrupt Summary status not detected 1b - Normal Interrupt Summary status detected
14 AIS	Abnormal Interrupt Summary Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: Bit 1: Transmit Process Stopped Bit 7: Receive Buffer Unavailable Bit 8: Receive Process Stopped Bit 10: Early Transmit Interrupt Bit 12: Fatal Bus Error Bit 13: Context Descriptor Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Abnormal Interrupt Summary status not detected 1b - Abnormal Interrupt Summary status detected
13 CDE	Context Descriptor Error This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one's descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Context Descriptor Error status not detected 1b - Context Descriptor Error status detected
12 FBE	Fatal Bus Error This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Fatal Bus Error status not detected 1b - Fatal Bus Error status detected
11 ERI	Early Receive Interrupt This bit when set indicates that the RxDMA has completed the transfer of packet data to the memory. In configurations supporting ERIC, ERIC = 0: This bit is set only after the Rx DMA has completely filled up a receive buffer with packet data. ERIC = 1: This bit is set after every burst transfer of data from the Rx DMA to the buffer. The setting of RI bit automatically clears this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Early Receive Interrupt status not detected 1b - Early Receive Interrupt status detected
10 ETI	Early Transmit Interrupt This bit when set indicates that the TxDMA has completed the transfer of packet data to the MTL TXFIFO memory. In configurations supporting ERIC: ETIC = 0: This bit is set only after the Tx DMA has transferred a complete packet to MTL. ETIC = 1: This bit is set after completion of (partial) packet data transfer from buffers in the Transmit descriptor in which IOC = 1. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Early Transmit Interrupt status not detected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Early Transmit Interrupt status detected
9 RWT	Receive Watchdog Timeout This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received. 0b - Receive Watchdog Timeout status not detected 1b - Receive Watchdog Timeout status detected
8 RPS	Receive Process Stopped This bit is asserted when the Rx process enters the Stopped state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Process Stopped status not detected 1b - Receive Process Stopped status detected
7 RBU	Receive Buffer Unavailable This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Buffer Unavailable status not detected 1b - Receive Buffer Unavailable status detected
6 RI	Receive Interrupt This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the Running state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Receive Interrupt status not detected 1b - Receive Interrupt status detected
5-3 —	Reserved
2 TBU	Transmit Buffer Unavailable This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPS0 field of the DMA_Debug_Status0 register explains the Transmit Process state transitions.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	To resume processing the Transmit descriptors, the application should do the following: Change the ownership of the descriptor by setting Bit 31 of TDES3. Issue a Transmit Poll Demand command. For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Buffer Unavailable status not detected 1b - Transmit Buffer Unavailable status detected
1 TPS	Transmit Process Stopped This bit is set when the transmission is stopped. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Process Stopped status not detected 1b - Transmit Process Stopped status detected
0 TI	Transmit Interrupt This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. 0b - Transmit Interrupt status not detected 1b - Transmit Interrupt status detected

70.5.1.121 Channel 1 Missed Frame Counter (DMA_CH1_MISS_FRAME_CNT)

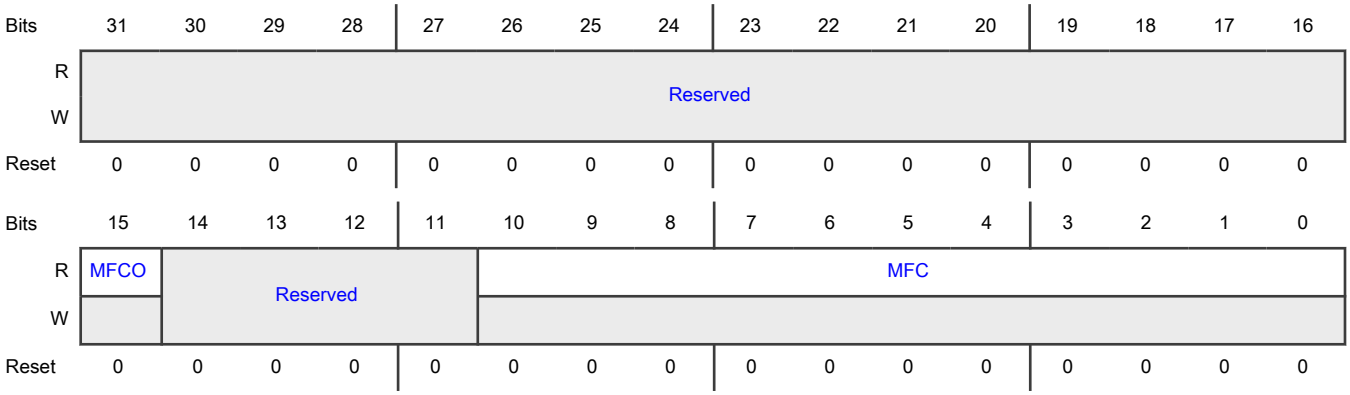
Offset

Register	Offset
DMA_CH1_MISS_FRAME_CNT	11E4h

Function

This register has the number of packet counter that got dropped by the DMA either due to Bus Error or due to programming RPF field in RX_CONTROL register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 MFCO	Overflow status of the MFC Counter When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event. 0b - Miss Frame Counter overflow not occurred 1b - Miss Frame Counter overflow occurred
14-11 —	Reserved
10-0 MFC	Dropped Packet Counters This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in DMA_CH${i}_Rx_Control register. The counter gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.

70.5.1.122 Channel 1 Receive ERI Counter (DMA_CH1_RX_ERI_CNT)

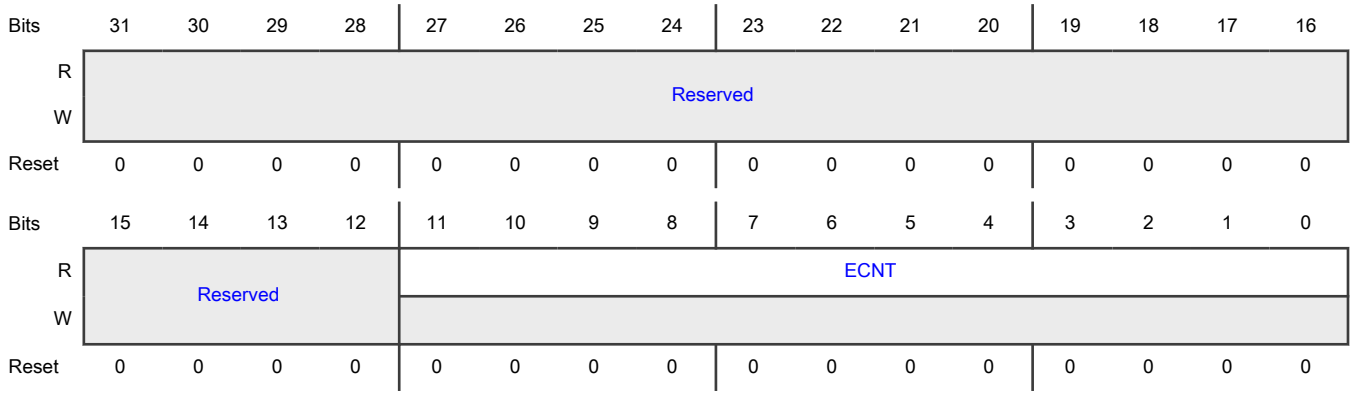
Offset

Register	Offset
DMA_CH1_RX_ERI_CNT	11ECh

Function

The Channel 1 Receive ERI Counter register provides the count of the number of times ERI was asserted.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-0 ECNT	ERI Counter When ERIC bit of RX_CONTROL register is set, this counter increments for burst transfer completed by the Rx DMA from the start of packet transfer. This counter will get reset at the start of new packet.

Chapter 71
Flexible Input/Output (FlexIO)

71.1 Chip-specific FlexIO information

Table 669. Reference links to related information

Topic	Related module	Reference
Full description	FlexIO	FlexIO
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See FLEXIO_TRIG0 registers in INPUTMUX

71.1.1 Module instances

This device has one instance of the FlexIO module, FLEXIO0.

71.1.2 FlexIO pin functions

FlexIO data input/output pin functions are mapped to various SoC pins and are selected by the Pin Function registers (see Signal multiplexing for more details). The table below lists the FlexIO data input/output pin functions mapped to the SoC-level pins.

NOTE

SoC pins are pinned out on all packages.

Table 670. FlexIO data input/output pin functions mapped to SoC-level pins

FlexIO pin	SoC pin	SoC pin	SoC pin
D0	P0_8	P0_16	
D1	P0_9	P0_17	
D2	P0_10	P0_18	
D3	P0_11	P0_19	
D4	P0_12	P0_20	
D5	P0_13	P0_21	
D6	P0_14	P0_22	
D7	P0_15	P0_23	
D8	P1_0	P2_0	P3_0
D9	P1_1	P2_1	P3_1
D10	P1_2	P2_2	P3_2
D11	P1_3	P2_3	P3_3

Table continues on the next page...

Table 670. FlexIO data input/output pin functions mapped to SoC-level pins (continued)

FlexIO pin	SoC pin	SoC pin	SoC pin
D12	P1_4	P2_4	P3_4
D13	P1_5	P2_5	P3_5
D14	P1_6	P2_6	P3_6
D15	P1_7	P2_7	P3_7
D16	P1_8	P2_8	P3_8
D17	P1_9	P2_9	P3_9
D18	P1_10	P2_10	P3_10
D19	P1_11	P2_11	P3_11
D20	P1_12	P4_12	P3_12
D21	P1_13	P4_13	P3_13
D22	P1_14	P4_14	P3_14
D23	P1_15	P4_15	P3_15
D24	P1_16	P4_16	P3_16
D25	P1_17	P4_17	P3_17
D26	P1_18	P4_18	P3_18
D27	P1_19	P4_19	P3_19
D28	P1_20	P4_20	P3_20
D29	P1_21	P4_21	P3_21
D30	P1_22	P4_22	P3_22
D31	P1_23	P4_23	P3_23

71.1.3 FlexIO external trigger connections

The FlexIO external trigger inputs come from various sources within the SoC and are selected by the FlexIO peripheral input muxes residing at INPUTMUX_BASE address plus offsets 0x6E0 – 0x6FC (see INPUTMUX chapter). The trigger input mux at 0x6E0 selects the external trigger input 0 source, the trigger input mux at 0x6E4 selects the external trigger input 1 source, and so on. The table below lists the triggers mapped to the FlexIO input sources.

Table 671. FlexIO external trigger connections

Number	Input Source
0	PIN_INT4
1	PIN_INT5
2	PIN_INT6
3	PIN_INT7
4	SCT_OUT5
5	SCT_OUT6

Table continues on the next page...

Table 671. FlexIO external trigger connections (continued)

Number	Input Source
6	SCT_OUT7
7	SCT_OUT8
8	SCT_OUT9
9	T0_MAT1
10	T1_MAT1
11	T2_MAT1
12	T3_MAT1
13	T4_MAT1
14	LPTMR0
15	LPTMR1
16	Reserved
17	GPIOINT_BMATCH
18	ADC0_tcomp0
19	ADC0_tcomp1
20	ADC0_tcomp2
21	ADC0_tcomp3
22	ADC1_tcomp0
23	ADC1_tcomp1
24	ADC1_tcomp2
25	ADC1_tcomp3
26	CMP0_OUT
27	CMP1_OUT
28	CMP2_OUT
29	PWM0_SM0_MUX_TRIG0
30	PWM0_SM0_MUX_TRIG1
31	PWM0_SM1_MUX_TRIG0
32	PWM0_SM1_MUX_TRIG1
33	PWM0_SM2_MUX_TRIG0
34	PWM0_SM2_MUX_TRIG1
35	PWM0_SM3_MUX_TRIG0
36	PWM0_SM3_MUX_TRIG1
37	PWM1_SM0_MUX_TRIG0

Table continues on the next page...

Table 671. FlexIO external trigger connections (continued)

Number	Input Source
38	PWM1_SM0_MUX_TRIG1
39	PWM1_SM1_MUX_TRIG0
40	PWM1_SM1_MUX_TRIG1
41	PWM1_SM2_MUX_TRIG0
42	PWM1_SM2_MUX_TRIG1
43	PWM1_SM3_MUX_TRIG0
44	PWM1_SM3_MUX_TRIG1
45	EVTG_OUT0A
46	EVTG_OUT0B
47	EVTG_OUT1A
48	EVTG_OUT1B
49	EVTG_OUT2A
50	EVTG_OUT2B
51	EVTG_OUT3A
52	EVTG_OUT3B
53	EXTTRIG_IN0
54	EXTTRIG_IN1
55	EXTTRIG_IN2
56	EXTTRIG_IN3
57	EXTTRIG_IN4
58	SINC_FILTER_CH0
59	SINC_FILTER_CH1
60	SINC_FILTER_CH2
61	SINC_FILTER_CH3
62	SINC_FILTER_CH4
63	LP_FLEXCOMM0_trig0
64	LP_FLEXCOMM0_trig1
65	LP_FLEXCOMM0_trig2
66	LP_FLEXCOMM1_trig0
67	LP_FLEXCOMM1_trig1
68	LP_FLEXCOMM1_trig2
69	LP_FLEXCOMM2_trig0

Table continues on the next page...

Table 671. FlexIO external trigger connections (continued)

Number	Input Source
70	LP_FLEXCOMM2_trig1
71	LP_FLEXCOMM2_trig2
72	LP_FLEXCOMM3_trig0
73	LP_FLEXCOMM3_trig1
74	LP_FLEXCOMM3_trig2
75	LP_FLEXCOMM3_trig3
76	WUU

71.2 Overview

FLEXIO is a highly configurable module that provides:

- Emulation of various serial or parallel communication protocols.
- Flexible 16-bit timers with support for various trigger, reset, enable, and disable conditions.
- Programmable logic blocks that allow the implementation of digital logic functions on-chip and configurable interaction of internal and external modules.
- Programmable state machine for offloading basic system control functions from the CPU.

71.2.1 Block diagram

The following diagram provides a high-level overview of the FLEXIO timer and shifter configuration.

FLEXIO uses shifters, timers, and external triggers to shift data into or out of FLEXIO. As shown in the block diagram, timers control the timing of this data shift. You can configure the timers to use generic timer functions, external triggers, or various other conditions to determine the control.

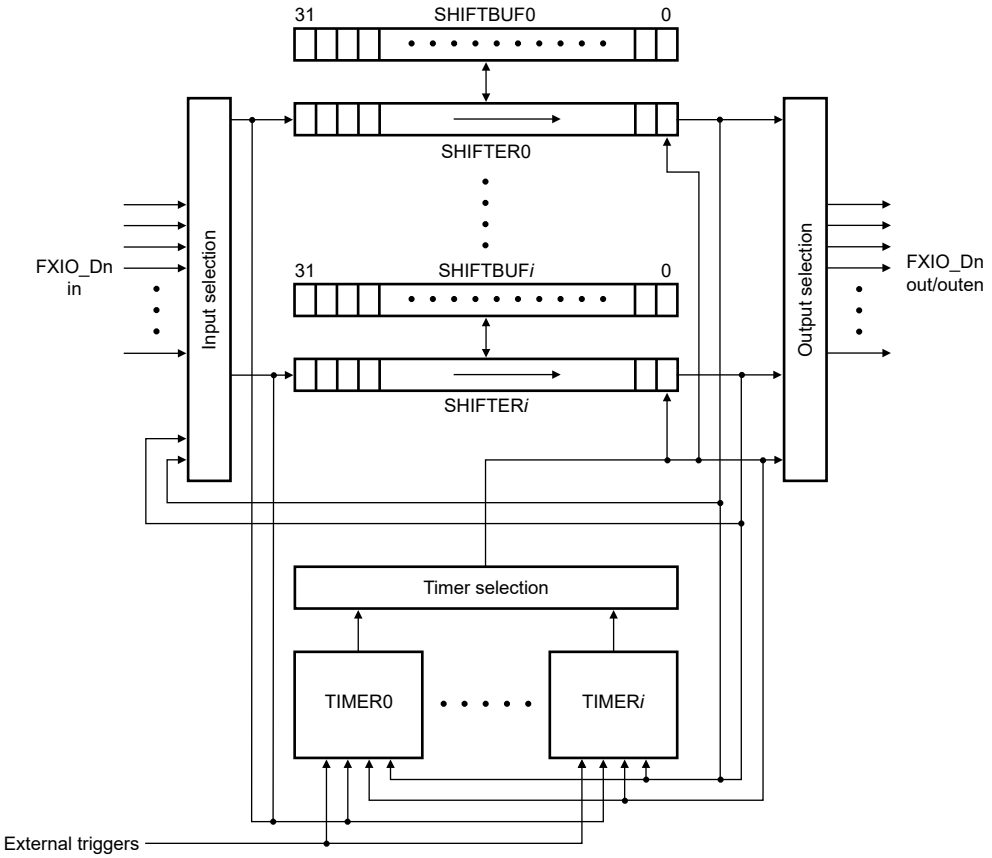


Figure 430. Block diagram

71.2.2 Features

- Array of 32-bit shift registers with transmit, receive, data match, logic, and state modes:
  - Double-buffered shifter operation for continuous data transfer
  - Shifter concatenation to support large transfer sizes
  - Automatic start and stop bit generation
  - 1, 2, 4, 8, 16, or 32 multi-bit shift widths for parallel interface support
  - Interrupt, DMA, or polled transmit and receive operation
- Highly flexible 16-bit timers with support for various internal or external triggers, reset, enable, and disable conditions:
  - Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during Stop mode
  - Programmable logic mode for integrating external digital logic functions on-chip, or combining pin, shifter, or timer functions to generate complex outputs
  - Programmable state machine for offloading basic system control functions from CPU, with support for up to eight states, eight outputs, and three selectable inputs per state
- Integrated general-purpose I/O registers and pin rising or falling edge interrupts to simplify software support
- Support for a wide range of protocols, including but not limited to:
  - UART

— I2C

— SPI

— I2S

— Camera IF

— Motorola 68K or Intel 8080 bus

— PWM or waveform generation

— Input-capture (pulse-edge interval measurement), such as SENT

71.3 Functional description

71.3.1 Shifter operation

Shifters are responsible for buffering and shifting data into or out of FLEXIO. The timer assigned to the shifter controls the timing of shift, load, and store events via SHIFTCTL*n*[TIMSEL]. Shifters are designed to support either DMA, interrupt, or polled operations. The following figure provides a detailed view of the shifter microarchitecture.

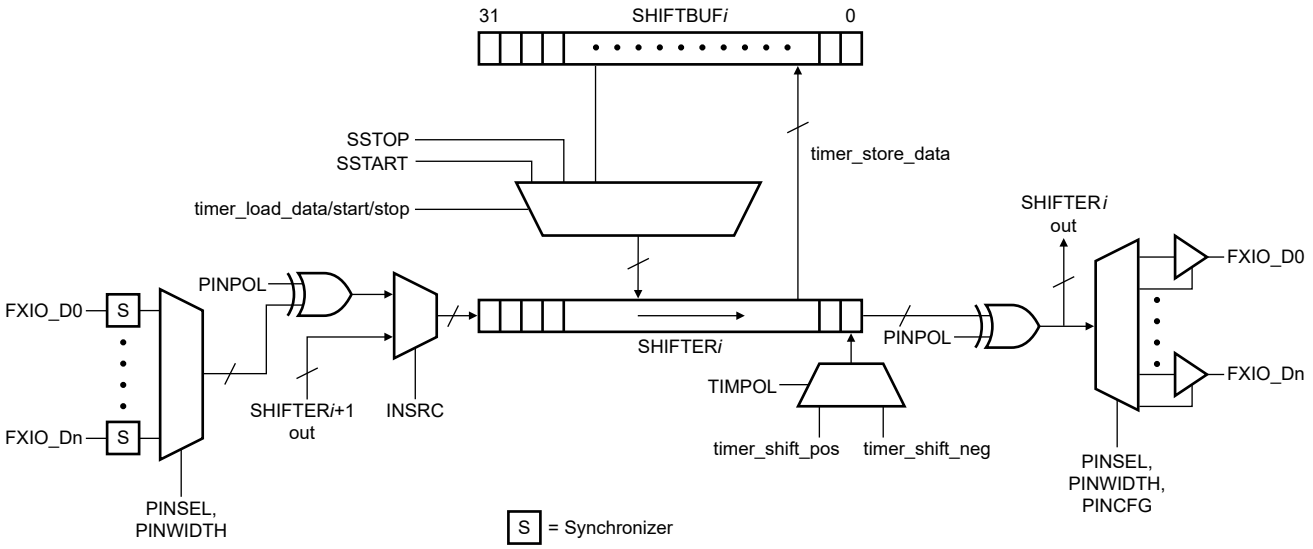


Figure 431. Shifter microarchitecture

71.3.1.1 Transmit mode

In Transmit mode (SHIFTCTL*n*[SMOD] = 010b), the shifter loads data from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) and shifts data out when the assigned timer signals a load event. An optional start and stop bit can be automatically loaded before or after SHIFTBUF register data by configuring either SHIFTCFG[SSTART] and TIMCFG[TSTART], or SHIFTCFG[SSTOP] and TIMCFG[TSTOP] in the shifter and timer.

NOTE

If a stop bit is enabled, the shifter immediately loads a stop bit when it is initially configured for Transmit mode.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when data has either been loaded from the SHIFTBUF register into the shifter or when the shifter is initially configured for Transmit mode. To clear the flag, write 1 or write new data to SHIFTBUF. In Transmit mode, write any value to the SHIFTBUF register to clear the corresponding shifter status flag, which is cleared regardless of what is writing to the register (DMA or interrupt), or the state of the DMA or interrupt enables. See the functional description of SHIFTSTAT[SSF] for information on how the flag is set and cleared for each mode.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when an attempt to load data from an empty SHIFTBUF register occurs (buffer underrun). Clear the flag by writing 1.

71.3.1.2 Receive mode

When the assigned timer signals a store event in Receive mode (SHIFTCTL*n*[SMOD] = 001b), the shifter shifts and stores data in Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). You can check for a start and stop bit before or after the shifter data is sampled by configuring either SHIFTCFG[SSTART] and TIMCFG[TSTART], or SHIFTCFG[SSTOP] and TIMCFG[TSTOP] in the shifter and timer.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when data is stored in the SHIFTBUF register from the shifter. To clear the flag, write 1 to or read the data from SHIFTBUF. Any read of the SHIFTBUF register clears the corresponding shifter status flag when the shifter is in Receive mode. The flag is cleared regardless of what is reading the register (DMA or interrupt) or the state of the DMA or interrupt enables. See the functional description of SHIFTSTAT[SSF] for information on how the flag is set or cleared for each mode.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set either when an attempt to store data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start or stop bit check. Write 1 to clear the flag.

71.3.1.3 Match Store mode

In Match Store mode (SHIFTCTL*n*[SMOD] = 100b), the shifter shifts data in, checks for a match result, and stores matched data in Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) when the assigned timer signals a store event. By configuring either SHIFTCFG[SSTART], TIMCFG[TSTART], and SHIFTCFG[SSTOP], or TIMCFG[TSTOP] in the shifter and timer, you can check for a start and stop bit before or after the shifter data is sampled. You can compare up to 16 bits of data using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when a match occurs and the matched data is stored in the SHIFTBUF register from the shifter. To clear the flag, read the matched data from the SHIFTBUF register or write 1 to the flag. Any read of the SHIFTBUF register clears the corresponding shifter status flag when the shifter is configured in Match Store mode. The flag is cleared regardless of what is reading the register (DMA or interrupt) or the state of the DMA or interrupt enables. See the functional description for SHIFTSTAT[SSF] to know how the flag is set or cleared for each mode.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when an attempt to store matched data into a full SHIFTBUF register occurs (buffer overrun), or when a mismatch occurs on a start or stop bit check. Write 1 to clear the flag.

71.3.1.4 Match Continuous mode

In Match Continuous mode (SHIFTCTL*n*[SMOD] = 101b), the shifter shifts data in and continuously checks for a match result whenever a shift event is signaled by the assigned timer. You can compare up to 16 bits of data using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when a match occurs. The flag clears automatically as soon as no match exists between the shifter data and Shifter Buffer (SHIFTBUF0 - SHIFTBUF7).

You cannot clear the flag by reading the SHIFTBUF register.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when a match occurs. To clear the flag, write 1 or perform a read from the SHIFTBUF register.

71.3.1.5 State mode

State mode enables you to implement any state machine with up to eight states, eight outputs, and three selectable inputs per state. This feature allows basic control functions to be offloaded from the CPU, which can remain in a low-power mode.

In State mode (SHIFTCTL*n*[SMOD] = 110b), when the shifter is selected by the current state pointer (SHIFTSTATE[STATE]), use the SHIFTBUF register to drive the output and compute the next state values. The following figure provides a detailed view of the shifter microarchitecture when configured for State mode.

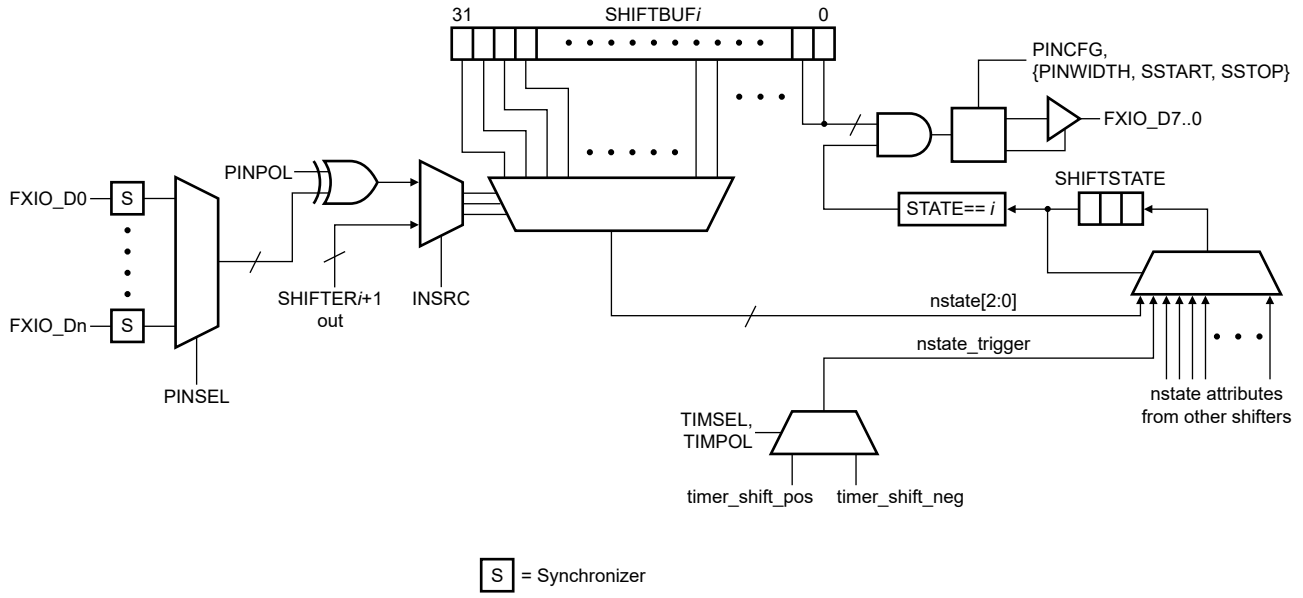


Figure 432. State microarchitecture

When the current state pointer selects a specific shifter (shifter *n*), output pins FXIO_D[7:0] are driven by SHIFTBUF*n*[31:24]; the configuration is defined by SHIFTCTL*n*[PINCFG]. Write 1 to Shifter Configuration (SHIFTCFG0 - SHIFTCFG7) {PWIDTH[3:0],SSTOP[1:0],SSTART[1:0]} to disable the output drive on pins FXIO_D[7:0] for state machine applications that require less than eight output pins.

Use the three input pins selected by SHIFTCTL*n*[PINSEL] and SHIFTBUF*n*[23:0] to compute the next state value.

NOTE

Each state can use a different set of three input pins.

The following table shows how the next state value is computed when the current state pointer is pointing to shifter *n*.

Table 672. Next state computation for SHIFTSTATE[STATE] = *n*

FXIO_D[PINSEL + 2]	FXIO_D[PINSEL + 1]	FXIO_D[PINSEL]	Next state value
0	0	0	SHIFTBUF*n*[2:0]
0	0	1	SHIFTBUF*n*[5:3]
0	1	0	SHIFTBUF*n*[8:6]
0	1	1	SHIFTBUF*n*[11:9]
...	...	...	...
1	1	1	SHIFTBUF*n*[23:21]

NOTE

You can configure other shifters and timers to drive the input pins of a given state, allowing you to create complex combinations of shifters and timers as needed. For example, the output of a shifter configured for Logic mode can be used to drive a state machine input.

The next state transition is triggered using the timer output selected by SHIFTCTL*n*[TIMSEL], with polarity controlled by SHIFTCTL*n*[TIMPOL].

NOTE

Each state can use a different timer to trigger each next state transition, allowing various internal or external trigger sources and clocking configurations to be used. See Timer section for more information.

The current state pointer defaults to shifter 0 at reset; however, you can write to select a different shifter for the initial state. If the current state pointer selects a shifter that is not configured for State mode, then outputs are not driven and the next state transition is never triggered.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when the shifter is selected by the current state pointer. The flag is cleared when the current state pointer is updated to a different shifter.

71.3.1.6 Logic mode

Logic mode enables you to implement a small amount of programmable digital logic within a FLEXIO shifter.

In Logic mode (SHIFTCTL*n*[SMOD] = 111b), use Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) to implement a 5-input, 32-bit programmable logic lookup table. The following figure provides a detailed view of shifter microarchitecture when configured for Logic mode.

Use the SHIFTBUF register to configure the lookup table for the four pin inputs. You can also use SHIFTER*n* to configure a feedback or delayed pin source as the fifth input to the lookup table.

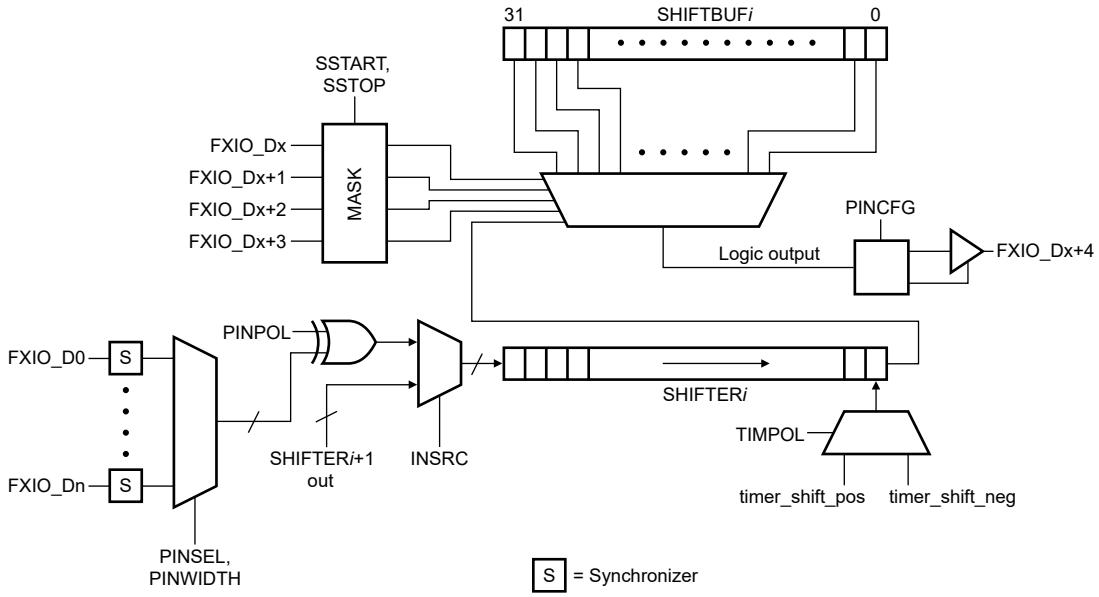


Figure 433. Logic microarchitecture

The lookup table is driven using four pin inputs (maskable using SHIFTCFG[SSTOP] and SHIFTCFG[SSTART]), plus one input from the internal shifter. It can be configured to drive an output pin using SHIFTCTL[PINCFG]. Pin inputs and outputs are fixed for each logic lookup table and are not selectable. The following table lists the logic output value selected by the lookup table for shifter *n*.

Table 673. Logic lookup table for shifter *n*

SHIFTER*n*[0]	FXIO_D[*x* + 3][1]	FXIO_D[*x* + 2]	FXIO_D[*x* + 1]	FXIO_D[*x*]	Logic output to FXIO_D[*x* + 4]
0	0	0	0	0	SHIFTBUF*n*[0]
0	0	0	0	1	SHIFTBUF*n*[1]

Table continues on the next page...

Table 673. Logic lookup table for shifter *n* (continued)

SHIFTER*n*[0]	FXIO_D[*x* + 3][1]	FXIO_D[*x* + 2]	FXIO_D[*x* + 1]	FXIO_D[*x*]	Logic output to FXIO_D[*x* + 4]
0	0	0	1	0	SHIFTBUF*n*[2]
0	0	0	1	1	SHIFTBUF*n*[3]
...	...	...	...	...	...
1	1	1	1	1	SHIFTBUF*n*[31]

1. for shifters *n* = 0...3, *x* = *n*

   for shifter *n* = 4...7, *x* = *n* + 4

To minimize output glitches, use SHIFTCFG*n*[SSTOP] and SHIFTCFG*n*[SSTART] to mask unused input pins. When these fields are 1, {SSTOP[1:0] and SSTART[1:0]} mask FXIO_D[*x* + 3]...FXIO_D[*x*] inputs respectively so that any transitions on these pins do not cause the logic output to glitch.

NOTE

You can configure other shifters and timers to drive the input pins of a given lookup table, allowing you to concatenate lookup tables or create complex combinations of shifters and timers as needed.

SHIFTCFG[PWIDTH] controls the number of delay stages introduced by the internal shifter input (SHIFTER*n*[0]). For example, when configured for a 1-bit shift (SHIFTCFG[PWIDTH] = 0), the internal shifter introduces a 32-shift clock delay before passing its input (selected by SHIFTCTL[PINSEL]) to the lookup table. When configured for a 32-bit shift (SHIFTCFG[PWIDTH] = 16...31), the internal shifter introduces a 1-shift clock delay to its input.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set whenever the output pin allocated to the logic lookup table has a value of 1 after being synchronized with the FLEXIO clock. The flag clears when the output pin has a value of 0. This also allows SHIFTSTAT[SSF] to be used as a trigger to a timer if needed.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when the output pin allocated to the logic lookup table is asserted. Clear the flag by writing 1 to it.

The Logic mode input pins, including pins driven by other shifters and timers, are synchronized with the FLEXIO functional clock before they are input to the programmable logic lookup table.

71.3.2 Timer operation

The FLEXIO 16-bit timers control the loading, shifting, and storing of the shift registers. The counters load the contents of the compare register and decrement down to zero on the FLEXIO clock. The counters can perform generic timer functions such as generating a clock, select output, or a PWM waveform. You can configure these timers to perform any of the following functions:

- Enable in response to a trigger, pin, or shifter condition.
- Decrement always or only on a trigger or pin edge.
- Reset in response to a trigger or pin condition.
- Disable on a trigger or pin condition or on a timer compare.

Timers can optionally include a start condition and a stop condition.

Although each timer operates independently, you can configure a timer to enable or disable at the same time as the previous timer (for example, timer 1 can enable or disable at the same time as timer 0) and a timer output can be used to trigger any other timer. The trigger used by each timer is configured independently as a timer output, shifter status flag, pin input, or an external trigger input. The trigger configuration is separate from pin configuration; you can perform it to configure input, output data, or output enable. See the chip-specific FLEXIO information for information on external trigger connections.

You must configure Timer Configuration (TIMCFG0 - TIMCFG7) before writing 1 to TIMCTL*n*[TIMOD].

71.3.2.1 Timer 8-bit Baud Counter mode

In 8-bit Baud Counter mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the baud rate of the shift clock and the upper 8 bits are used to configure the number of shift clock edges in the transfer. When the lower 8 bits decrement to zero, the timer output is toggled and the lower 8 bits reload from the compare register. The upper 8 bits decrement when the lower 8 bits become zero and decrement.

NOTE

A timer reset event in 8-bit Baud Counter mode only resets the lower 8-bit counter. The upper 8-bit counter is not affected and can decrement if the timer reset is configured to update the state of the timer output, which toggles as a result of the timer reset event.

A timer compare event occurs when the upper 8 bits equal zero and decrement. The timer status flag is set on a timer compare event.

71.3.2.2 Timer 8-bit High PWM mode

In 8-bit High PWM mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the timer output high period and the upper 8 bits are used to configure the timer output low period. The lower 8 bits decrement when the output is high. When the lower 8 bits become zero and decrement, the timer output is cleared and the lower 8 bits are reloaded from the compare register. The upper 8 bits decrement when the output is low. When the upper 8 bits become zero and decrement, the timer output is set and the upper 8 bits are reloaded from the compare register.

A timer compare event occurs when the upper 8 bits become zero and decrement. The timer status flag is set on a timer compare event.

71.3.2.3 Timer 16-bit Counter mode

In 16-bit Counter mode, you can use the 16-bit counter to configure either the baud rate of the shift clock (for example, TIMDEC[1:0] ≠ 10 or 11) or the number of shift clock edges in the transfer (for example, TIMDEC[1:0] = 10 or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer compare event.

71.3.2.4 Timer 16-bit Counter Disable mode

In 16-bit Counter Disable mode, the 16-bit counter can be used to configure either the baud rate of the shift clock (for example, TIMDEC[1:0] ≠ 10 or 11) or the number of shift clock edges in the transfer (for example, TIMDEC[1:0] = 10 or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer disable event.

71.3.2.5 Timer 8-bit Word Counter mode

In 8-bit Word Counter mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the number of shift clock edges in each word and the upper 8 bits are used to configure the number of words in the transfer. When the lower 8 bits decrement to zero, the timer output is toggled and the lower 8 bits reload from the compare register. The upper 8 bits only decrement when the lower 8 bits become zero and decrement.

A timer compare event occurs when the lower 8 bits become zero and decrement. The timer status flag is set when the upper 8 bits become zero and decrement.

71.3.2.6 Timer 8-bit Low PWM mode

In 8-bit Low PWM mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the timer output low period and the upper 8 bits are used to configure the timer output high period. The lower 8 bits decrement when the output is low. When the lower 8 bits become zero and decrement, the timer output is set and the lower 8 bits are reloaded from the compare

register. The upper 8 bits decrement when the output is high. When the upper 8 bits become zero and decrement, the timer output is cleared and the upper 8 bits are reloaded from the compare register.

A timer compare event occurs when the upper 8 bits become zero and decrement. The timer status flag is set on a timer compare event.

71.3.2.7 Timer enable and start functions

The following events occur when you configure TIMCTL*n*[TIMOD] for the desired mode and the condition configured by the timer enable (TIMCFG*n*[TIMENA]) is detected. When TIMCTL*n*[ONETIM] is 1, the timer status flag must be clear to generate a timer enable event; otherwise, the timer enable event is blocked. You can use this to enforce software intervention after each timer iteration:

- The timer counter loads the current value of the compare register and starts decrementing, as configured by TIMCFG*n*[TIMDEC].
- The timer output may update to its initial state depending on the configuration of TIMCFG*n*[TIMOUT]. Shifters that are controlled by this timer do not see this as a rising edge on the timer shift clock.
- Transmit shifters controlled by this timer either output their start bit value or load the shift register from the shift buffer and output the first bit, as configured by SHIFTCFG*n*[SSTART].

If the timer start bit is enabled, the timer counter reloads with the compare register on the first rising edge of the shift clock after the timer starts decrementing. If there is no falling edge on the shift clock before the first rising edge (for example, when TIMCFG*n*[TIMOUT] = 1), a shifter that is configured to shift on the falling edge and load on the first shift does not load correctly.

71.3.2.8 Timer decrement and reset functions

The timer generates the timer output and timer shift clock depending on the fields, TIMCTL*n*[TIMOD] and TIMCFG*n*[TIMDEC]. The shifter clock is either equal to the timer output (when TIMCFG*n*[TIMDEC] ≠ 10 or 11) or equal to the decrement clock (when TIMCFG*n*[TIMDEC] = 10 or 11). If you configure TIMCFG*n*[TIMDEC] to decrement from a pin or trigger, the timer decrements on both rising and falling edges.

If a timer is configured to decrement on the FLEXIO functional clock divided by 16 or 256 (when TIMCFG*n*[TIMDEC] = 100 or 101), then a common prescaler that is shared by all timers is used to generate the two divide ratios. This prescaler is reset when all timers are either idle or configured not to use the prescaler (TIMCFG*n*[TIMDEC] ≠ 100 or 101).

If you configure the timer to reset as determined by TIMCFG*n*[TIMRST], then the timer counter loads the current value of the compare register again. You can configure the timer output and timer shift clock to update on timer reset, as configured by TIMCFG*n*[TIMOUT]. If the time output toggles as a result of the timer reset, this can result in a timer shift clock edge. In 8-bit Baud Counter mode, this also decrements the upper 8 bits of the counter.

In general, when the timer counter decrements to zero, a timer compare event is triggered. The timer compare event causes:

- The timer counter to load the contents of the timer compare register.
- The timer output to toggle.
- Any configured transmit shift registers to load.
- Any configured receive shift registers to store.

Depending on the timer mode, the timer status flag may also be set.

71.3.2.9 Timer disable and stop functions

When the timer is configured to add a stop bit on each compare, the following additional events occur:

- Transmit shifters controlled by this timer output their stop bit value (if configured by SHIFTCFG*n*[SSTOP]).
- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by SHIFTCFG*n*[SSTOP].

- The timer counter reloads the current value of the compare register on the first rising edge of the shifter clock after the compare.

If you configure the timer to insert a stop bit on each compare, you must configure the transmit shifters to load on the first shift.

When the condition configured by timer disable (TIMCFG*n*[TIMDIS]) is detected, the following events occur:

- Timer counter reloads the current value of the compare register and starts decrementing as configured by TIMCFG*n*[TIMDEC].
- Timer output clears. Shifters that are controlled by this timer do not see this as a falling edge on the timer shift clock, but can generate a shift event if the timer shift clock otherwise generates one.
- Transmit shifters controlled by this timer output their stop bit value (if configured by SHIFTCFG*n*[SSTOP]).
- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by SHIFTCFG*n*[SSTOP].

If the timer stop bit is enabled, the timer counter continues decrementing until the next rising edge of the shift clock is detected, at which point it finishes decrementing. Although the timer output is forced low during the stop bit, the timer shift clock can toggle during the stop bit. The timer output does not generate shift events during the stop bit.

A timer enable condition can be detected in the same cycle as a timer disable condition (if timer stop bit is disabled), or on the first rising edge of the shift clock after the disable condition (if stop bit is enabled). When TIMCTL*n*[ONETIM] is 1, the timer status flag must be clear before the next timer enable condition is detected. When the timer is in the stop state condition, receive shift registers with stop bit enabled store the contents of the shift register into the shift buffer and verify the state of the input data on the configured shift edge. If there is no configured edge between the timer disable and the next rising edge of the shift clock, then the final store and verify do not occur.

71.3.3 Pin operation

The pin configuration for each timer and shifter can be set to use any FLEXIO pin with either polarity. You can configure each timer and shifter as an input, output data, output enable, or bidirectional output. A pin configured for output enable can be used as an open drain (with inverted polarity because the output enable assertion causes logic zero to be output on the pin) or to control the enable on the bidirectional output. You can configure any timer or shifter to control the output enable for a pin where the bidirectional output data is driven by another timer or shifter.

71.3.3.1 Parallel interface

You can configure shifters to use multiple FLEXIO pins simultaneously by using SHIFTCFG*n*[PWIDTH], which is used to configure the following settings of a shifter:

1. Number of bits shifted per shift clock.
2. Number of pins driven by the shifter per shift clock (only on shifters supporting parallel transmit—that is, SHIFTER0 and SHIFTER4.)
3. Number of pins sampled by the shifter per shift clock (only on shifters supporting parallel receive—that is, SHIFTER3 and SHIFTER7.)

When configured for parallel shift, either 4, 8, 16, or 32 bits can be shifted on every shift clock. If an adjacent shifter is selected as the input source (SHIFTCFG*n*[INSRC] = 1), the least significant 4, 8, 16, or 32 bits from the adjacent shifter are sampled on each shift clock.

For shifters supporting parallel receive (SHIFTER3, SHIFTER7), you can configure the shifter to sample multiple pins (INSRC = 0), with PWIDTH and PINSEL selecting the pins as FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL].

NOTE

If PWIDTH is less than the number of bits being shifted on each shift clock, then the most significant bits are masked with 0. For example, if PINSEL = 7 and PWIDTH = 6, then SHIFTER[31:24] samples {0,0,FXIO_D[12:7]} on each shift clock.

For shifters supporting parallel transmit (SHIFTER0, SHIFTER4), you can configure the shifter to drive multiple pins using SHIFTCTL*n*[PINCFG], with PWIDTH and PINSEL selecting the pins as follows: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL].

NOTE

If PWIDTH is less than the number of bits being shifted on each shift clock, then the most significant pins are not driven. For example, if PINSEL = 7 and PWIDTH = 6, then SHIFTER[5:0] drives only FXIO_D[12:7] on each shift clock.

71.3.3.2 Pin synchronization

When you configure a pin as an input (this includes a timer trigger configured as a pin input), the input signal is first synchronized with the FLEXIO clock before a timer or shifter could use the signal. This introduces a small latency of 0.5–1.5 FLEXIO clock cycles when using an external pin input to generate an output or control a shifter. This sets the maximum setup time at 1.5 FLEXIO clock cycles.

If an input is used by more than one timer or shifter, then the synchronization occurs once to ensure any edge is seen on the same cycle by all timers and shifters using that input.

NOTE

FLEXIO pins are also connected internally. Configuring a FLEXIO shifter or timer to output data on an unused pin makes an internal connection that allows other shifters and timers to use this pin as an input. This allows a shifter output to trigger a timer or a timer output to be shifted into a shifter. This path is also synchronized with the FLEXIO clock and therefore incurs a one-cycle latency.

When using a pin input as a timer trigger, timer clock, or shifter data input, the following synchronization delays occur:

- 0.5–1.5 FLEXIO clock cycles for an external pin
- One FLEXIO clock cycle for an internally driven pin

See Application information for timing considerations such as output valid time and input setup time for specific applications (SPI controller, SPI target, I2C controller, I2S controller, and I2S target).

71.3.3.3 Pin override

You can change the state of any FLEXIO pin at any time. Pin Output Enable (PINOUTE) configures any pin as an output and drives that pin with the value in Pin Output Data (PINOUTD).

Alias registers for PINOUTE and data registers also exist. Writing a logic 1 to an alias register updates the corresponding register fields in both PINOUTE and PINOUTD as follows:

- Pin Output Disable (PINOUTDIS) clears Pin Output Enable (PINOUTE) and Pin Output Data (PINOUTD).
- Pin Output Clear (PINOUTCLR) sets Pin Output Enable (PINOUTE) and clears Pin Output Data (PINOUTD).
- Pin Output Set (PINOUTSET) sets Pin Output Enable (PINOUTE) and Pin Output Data (PINOUTD).
- Pin Output Toggle (PINOUTTOG) sets Pin Output Enable (PINOUTE) and toggles Pin Output Data (PINOUTD).

71.3.3.4 Pin interrupt

You can read the state of any FLEXIO pin at any time and also configure any pin to set a status flag when either a rising or falling edge is detected on that pin. Additionally, you can configure the pin status flag to generate an interrupt.

71.3.4 Low-power modes

FLEXIO remains functional during low-power modes, if CTRL[DOZEN] is 0 and the FLEXIO functional clock remains enabled.

The exception to this is in PD mode. In this case, CTRL[DOZEN] is ignored and FLEXIO waits for all timers to complete any pending operation before acknowledging PD mode entry.

71.3.5 Debug mode

FLEXIO remains functional in Debug mode, provided the value of CTRL[DBGE] is 1.

71.3.6 Clocking

Table 674. FLEXIO clocks

Clock	Description
Functional clock	Is asynchronous to the bus clock and can remain enabled in low-power modes. You must enable the FLEXIO functional clock before accessing any of the FLEXIO registers. Provided the FLEXIO functional clock is at least equal to the bus clock, you can configure CTRL[FASTACC] to support fast register accesses.
Bus clock	Is used only for bus accesses to the control and configuration registers.

71.3.7 Reset

Table 675. FLEXIO reset types

Reset	Description
Chip reset	Resets the FLEXIO logic and registers to their default states on chip reset.
Software reset	Resets, using CTRL[SWRST], all logic and registers to their default states, except for the Control register.

71.3.8 Interrupts and DMA requests

The following table shows the status flags that generate the FLEXIO interrupt and DMA requests.

Table 676. FLEXIO interrupts and DMA requests

Flag	Description	Interrupt	DMA request	Low-power wake-up
SHIFTSTAT[SSF]	Shifter status flag	Y	Y	Y
SHIFTERR[SEF]	Shifter error flag	Y	N	Y
TIMSTAT[TSF]	Timer status flag	Y	Y	Y
PINSTAT[PSF]	Pin status flag	Y	N	Y
TRGSTAT[ETSF]	External trigger status flag	Y	N	Y

71.3.9 Peripheral triggers

The connection between FLEXIO peripheral triggers and other peripherals is device-specific.

71.3.9.1 Output triggers

Each FLEXIO timer generates an output trigger equal to the timer output. The output trigger is not affected by the timer pin polarity configuration.

71.3.9.2 Input trigger

FLEXIO supports multiple external trigger inputs that can be used to trigger one or more FLEXIO timers. If a rising edge is detected on an external trigger when FLEXIO is enabled, then the external trigger status flag is set. The external triggers are synchronized to the FLEXIO functional clock and must assert for at least two cycles of the FLEXIO functional clock to be sampled correctly.

71.4 External signals

Table 677. External signals

Signal	Description	Direction
FXIO_D*n* (*n* = 0...31)	Bidirectional FLEXIO shifter and timer pin	Input or output

71.5 Initialization

Perform the following procedure to initialize FLEXIO registers:

1. Enable FLEXIO by writing 1 to CTRL[FLEXEN].
2. Configure shift registers for the given application. It is recommended to write to Shifter Configuration (SHIFTCFG0 - SHIFTCFG7) before writing to the corresponding register, Shifter Control (SHIFTCTL0 - SHIFTCTL7).
3. Configure timer registers for the given application. It is recommended to write to Timer Compare (TIMCMP0 - TIMCMP7) and Timer Configuration (TIMCFG0 - TIMCFG7) before writing to the corresponding register, Timer Control (TIMCTL0 - TIMCTL7).
4. Enable interrupts and/or DMA requests, as appropriate, for the given application.
5. Write transmit data to initiate a transfer (depending on the given application).

71.6 Application information

This section provides examples for a variety of FLEXIO module applications. See FLEXIO register descriptions for more information.

71.6.1 UART transmit

UART transmit can be supported using one timer, one shifter, and one pin (two pins, if supporting CTS). The start and stop bit insertion is handled automatically, and multiple transfers are supported using the DMA controller. The timer status flag is used to indicate when the stop bit of each word is transmitted.

Break and idle characters require software intervention. Before transmitting a break or idle character, you must modify SHIFTCFG*n*[SSTART] and SHIFTCFG*n*[SSTOP] to transmit the required state, and the data to transmit must equal FFh or 00h. Supporting a second stop bit requires the stop bit to be inserted into the data stream using software (and increasing the number of bits to transmit). When performing byte writes to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) (or Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) for transmitting MSB first), the rest of the register remains unaltered. This allows an address mark bit or additional stop bit to remain undisturbed.

NOTE

FLEXIO does not support automatic insertion of parity bits.

Table 678. UART transmit configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0032h	Configure start bit of 0 and stop bit of 1.
SHIFTCTL*n*	0003_0002h	Configure transmit using timer 0 on the positive edge of clock with output data on pin 0. You can configure the PINPOL field to invert output data, or support open-drain by writing 1h to the PINPOL and PINCFG fields.

Table continues on the next page...

Table 678. UART transmit configuration (continued)

Register	Value	Configuration
TIMCMP*n*	0000_0F01h	Configure 8-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG*n*	0000_2222h	Configure start bit, stop bit, enable on trigger asserted and disable on compare. You can support CTS by configuring the TIMENA field as 3h.
TIMCTL*n*	01C0_0001h	Configure the dual 8-bit counter using the shifter 0 status flag as an inverted internal trigger source. To support CTS, configure the PINSEL (for pin 1) and PINPOL fields as 1h.
SHIFTBUF*n*	Data to transmit	Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer. Use the shifter status flag to indicate when data can be written using an interrupt or a DMA request. Write to SHIFTBUFBBS[7:0] instead to support MSB first transfer.

The following table shows an alternative configuration that supports slower baud rates. This configuration requires two timers.

Table 679. UART transmit configuration for slow baud rate

Register	Value	Configuration
SHIFTCFG*n*	0000_0032h	Configure start bit of 0 and stop bit of 1.
SHIFTCTL*n*	0003_0002h	Configure transmit using timer 0 on the positive edge of clock with output data on pin 0. Invert output data by writing 1 to the PINPOL field. Support open-drain by configuring the PINPOL and PINCFG fields as 1h.
TIMCMP*n*	0000_000Fh	Configure for 8-bit transfer, and configure TIMCMP[15:0] as (number of bits × 2) - 1.
TIMCFG*n*	0030_2622h	Configure start bit, stop bit, enable on trigger rising edge, decrement on trigger and disable on compare.
TIMCTL*n*	0740_0003h	Configure the 16-bit counter using the timer 1 output as an internal trigger source.
TIMCMP(*n* + 1)	0000_0001h	Configure baud rate of divide by 4 of the FLEXIO clock, and configure

Table continues on the next page...

Table 679. UART transmit configuration for slow baud rate (continued)

Register	Value	Configuration
		TIMCMP[15:0] as (baud rate divider ÷ 2) - 1.
TIMCFG(*n* + 1)	0000_1200h	Configure enable on trigger asserted and disable on timer 0 disable. You can configure the TIMEN field as 3h to support CTS.
TIMCTL(*n* + 1)	01C0_0003h	Configure the 16-bit counter using the shifter 0 status flag as an inverted internal trigger source. You can support CTS by configuring the PINSEL (for pin 1) and PINPOL fields as 1h.
SHIFTBUF*n*	Data to transmit	Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer. Use the shifter status flag to indicate when data can be written using an interrupt or a DMA request. Write to SHIFTBUFBBS[7:0] instead to support MSB first transfer.

71.6.2 UART receive

UART receive can be supported using one timer, one shifter, and one pin (two timers and two pins, if supporting RTS). The start and stop bit verification is handled automatically and multiple transfers are supported using the DMA controller. The timer status flag is used to indicate when the stop bit of each word is received.

FLEXIO does not support triple voting of the received data, which is sampled only once in the middle of each bit. You can use a timer to implement a glitch filter on the incoming data and a different timer to detect an idle line of programmable length. Break characters cause the error flag to set, and the shifter buffer register returns 00h.

NOTE

FLEXIO does not support automatic verification of parity bits.

Table 680. UART receiver configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0032h	Configure start bit of 0 and stop bit of 1.
SHIFTCTL*n*	0080_0001h	Configure receive using timer 0 on the negative edge of clock with input data on pin 0. You can invert input data by writing 1 to the PINPOL field.
TIMCMP*n*	0000_0F01h	Configure 8-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG*n*	0204_2422h	Configure start bit, stop bit, enable on pin positive edge and disable on

Table continues on the next page...

Table 680. UART receiver configuration (continued)

Register	Value	Configuration
		compare. Enable resynchronization to received data with TIMOUT = 2h and TIMRST = 4h.
TIMCTL*n*	0000_0081h	Configure the dual 8-bit counter using the inverted pin 0 input.
SHIFTBUF*n*	Data to receive	You can read received data from SHIFTBUFBYS[7:0]. Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from SHIFTBUFBIS[7:0] instead to support MSB first transfer.

The UART receiver with RTS configuration uses a second timer to generate the RTS output. RTS asserts when the start bit is detected and negates when the data is read from the shifter buffer register. If no start bit is detected when the RTS is asserted, the received data is ignored.

Table 681. UART receiver with RTS configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0032h	Configure start bit of 0 and stop bit of 1.
SHIFTCTL*n*	0080_0001h	Configure receive using timer 0 on the negative edge of clock with input data on pin 0. Invert input data by writing 1 to the PINPOL field.
TIMCMP*n*	0000_0F01h	Configure 8-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG*n*	0204_2522h	Configure start bit, stop bit, enable on pin positive edge with trigger asserted and disable on compare. Enable resynchronization to received data with TIMOUT = 2h and TIMRST = 4h.
TIMCTL*n*	02C0_0081h	Configure dual 8-bit counter using the inverted pin 0 input. Trigger is internal using the inverted pin 1 input.
TIMCMP(*n* + 1)	0000_FFFFh	Never compare.
TIMCFG(*n* + 1)	0030_6100h	Enable on timer *n* enable and disable on the trigger falling edge. Decrement on trigger to ensure no compare.
TIMCTL(*n* + 1)	0143_0003h	Configure 16-bit counter and output on pin 1. Trigger is internal using the shifter 0 flag.

Table continues on the next page...

Table 681. UART receiver with RTS configuration (continued)

Register	Value	Configuration
SHIFTBUF*n*	Data to receive	You can read received data using SHIFTBUFBYS[7:0]. Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from SHIFTBUFBIS[7:0] instead to support MSB first transfer.

71.6.3 SPI controller

SPI Controller mode can be supported using two timers, two shifters, and four pins. Using the DMA controller, either CPHA = 0 or CPHA = 1 and transfers can be supported. For CPHA = 1, the chip select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The stop bit is used to guarantee a minimum of one clock cycle between the target chip select negating and before the next transfer. To initiate each transfer, either the core or DMA writes to the transmit buffer.

NOTE

Because of synchronization delays, the setup time for the serial input data is 1.5 FLEXIO clock cycles. This means the maximum baud rate is divide by 4 of the FLEXIO clock frequency.

Table 682. SPI controller (CPHA = 0) configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0000h	Start and stop bit disabled.
SHIFTCTL*n*	0083_0002h	Configure transmit using timer 0 on the negative edge of clock with output data on pin 0.
SHIFTCFG(*n* + 1)	0000_0000h	Start and stop bit disabled.
SHIFTCTL(*n* + 1)	0000_0101h	Configure receive using timer 0 on the positive edge of clock with input data on pin 1.
TIMCMP*n*	0000_3F01h	Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG*n*	0100_2222h	Configure start bit, stop bit, enable on trigger high and disable on compare; initial clock state is logic 0.
TIMCTL*n*	01C3_0201h	Configure dual 8-bit counter using the pin 2 output (shift clock), with shifter 0 flag as the inverted trigger. Write 1 to the PINPOL field to invert the output shift clock.
TIMCMP(*n* + 1)	0000_FFFFh	Never compare.

Table continues on the next page...

Table 682. SPI controller (CPHA = 0) configuration (continued)

Register	Value	Configuration
TIMCFG(*n* + 1)	0000_1100h	Enable when timer 0 is enabled and disable when timer 0 is disabled.
TIMCTL(*n* + 1)	0003_0383h	Configure 16-bit counter (never compare) using the inverted pin 3 output as target select.
SHIFTBUF*n*	Data to transmit	You can write transmit data to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.
SHIFTBUF(*n* + 1)	Data to receive	Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.

Table 683. SPI controller (CPHA = 1) configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0021h	Start bit loads data on first shift.
SHIFTCTL*n*	0003_0002h	Configure transmit using timer 0 on the positive edge of clock with output data on pin 0.
SHIFTCFG(*n* + 1)	0000_0000h	Start and stop bit disabled.
SHIFTCTL(*n* + 1)	0080_0101h	Configure receive using timer 0 on the negative edge of clock with input data on pin 1.
TIMCMP*n*	0000_3F01h	Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits x 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG*n*	0100_2222h	Configure start bit, stop bit, enable on trigger high and disable on compare; initial clock state is logic 0.
TIMCTL*n*	01C3_0201h	Configure dual 8-bit counter using pin 2 output (shift clock), with the shifter 0 flag as the inverted trigger. Write 1 to the

Table continues on the next page...

Table 683. SPI controller (CPHA = 1) configuration (continued)

Register	Value	Configuration
		PINPOL field to invert the output shift clock, and set the TIMDIS field as 3 to keep target select asserted for as long as there is data in the transmit buffer.
TIMCMP(*n* + 1)	0000_FFFFh	Never compare.
TIMCFG(*n* + 1)	0000_1100h	Enable when timer 0 is enabled and disable when timer 0 is disabled.
TIMCTL(*n* + 1)	0003_0383h	Configure 16-bit counter (never compare) using inverted pin 3 output (as target select).
SHIFTBUF*n*	Data to transmit	Transmit data can be written to SHIFTBUF. Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.
SHIFTBUF(*n* + 1)	Data to receive	Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.

71.6.4 SPI target

SPI Target mode can be supported using one timer, two shifters, and four pins. Either CPHA = 0 or CPHA = 1 can be supported and transfers can be supported using the DMA controller. For CPHA = 1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

You must write the transmit data to the transmit buffer register before the external target select asserts; otherwise, the shifter error flag is set.

NOTE

Because of synchronization delays, the output valid time for the serial output data is 2.5 FLEXIO clock cycles. This means the maximum baud rate is divide by 6 of the FLEXIO clock frequency.

Table 684. SPI target (CPHA = 0) configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0000h	Start and stop bit disabled.
SHIFTCTL*n*	0083_0002h	Configure transmit using timer 0 on the falling edge of shift clock with output data on pin 0.

Table continues on the next page...

Table 684. SPI target (CPHA = 0) configuration (continued)

Register	Value	Configuration
SHIFTCFG(*n* + 1)	0000_0000h	Start and stop bit disabled.
SHIFTCTL(*n* + 1)	0000_0101h	Configure receive using timer 0 on the rising edge of shift clock with input data on pin 1.
TIMCMP*n*	0000_003Fh	Configure 32-bit transfer. Set TIMCMP[15:0] as (number of bits × 2) - 1.
TIMCFG*n*	0120_0600h	Configure enable on trigger rising edge. Initial clock state is logic 0 and decrements on pin input.
TIMCTL*n*	06C0_0203h	Configure 16-bit counter using pin 2 input (shift clock), with pin 3 input (target select) as the inverted trigger.
SHIFTBUF*n*	Data to transmit	Transmit data can be written to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.
SHIFTBUF(*n* + 1)	Data to receive	Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.

Table 685. SPI target (CPHA = 1) configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0001h	Shifter configured to load on first shift and stop bit disabled.
SHIFTCTL*n*	0003_0002h	Configure transmit using timer 0 on rising edge of shift clock with output data on pin 0.
SHIFTCFG(*n* + 1)	0000_0000h	Start and stop bit disabled.
SHIFTCTL(*n* + 1)	0080_0101h	Configure receive using timer 0 on falling edge of shift clock with input data on pin 1.

Table continues on the next page...

Table 685. SPI target (CPHA = 1) configuration (continued)

Register	Value	Configuration
TIMCMP*n*	0000_003Fh	Configure 32-bit transfer. Set TIMCMP[15:0] as (number of bits × 2) - 1).
TIMCFG*n*	0120_6602h	Configure start bit, enable on trigger rising edge, disable on trigger falling edge. Initial clock state is logic 0 and decrements on pin input.
TIMCTL*n*	06C0_0203h	Configure 16-bit counter using pin 2 input (shift clock), with pin 3 input (target select) as the inverted trigger.
SHIFTBUF*n*	Data to transmit	Transmit data can be written to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.
SHIFTBUF(*n* + 1)	Data to receive	Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer.

71.6.5 I2C controller

I2C Controller mode can be supported using two timers, two shifters, and two pins. One timer is used to generate the SCL output and another one is used to control the shifters. The two shifters that are used to transmit and receive for every word, when receiving the transmitter, must transmit FFh to 3-state the output. FLEXIO inserts a stop bit after every word to generate and verify the ACK or NACK. FLEXIO waits for the first write to the transmit data buffer before enabling SCL generation. Data transfers can be supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The first timer generates the bit clock for the entire packet (start to repeated start or stop condition), so you must program the compare register with the total number of clock edges in the packet (minus one). The timer supports clock stretching using the reset counter when pin is equal to output. However, this increases both the clock high and clock low periods by at least one FLEXIO clock cycle each. The second timer uses the SCL input pin to control the transmit and receive shift registers. This enforces an SDA data hold time by an extra two FLEXIO clock cycles.

Both the transmit and receive shifters must be serviced for each word in the transfer. The transmit shifter must transmit FFh when receiving, and the receive shifter returns the data present on the SDA pin. The transmit shifter loads one additional word on the last falling edge of the SCL pin. When generating a stop condition or a repeated start condition, this word must be 00h and FFh, respectively. During the last word of a controller-receiver transfer, you must set the transmit SHIFTCFG*n*[SSTOP] field to generate a NACK.

The receive shift register asserts an error interrupt if a NACK is detected, but you are responsible for generating the stop or repeated start condition. If a NACK is detected during controller-transmit, the interrupt routine must immediately write 00h (when generating a stop condition) or FFh (when generating a repeated start condition) to the transmit shifter register. You must wait for

the next rising edge on SCL before disabling both timers. The transmit shifter must be disabled after the setup delay for a repeated start or stop condition.

NOTE

Because of synchronization delays, the data valid time for the transmit output is two FLEXIO clock cycles. This means the maximum baud rate is divide by 6 of the FLEXIO clock frequency.

To guarantee SDA hold time, the I2C controller data valid is delayed by two cycles because the clock output is passed through a synchronizer before clocking the transmit or receive shifter. Because the SCL output is synchronous with FLEXIO clock, the synchronization delay is one cycle, and then an additional cycle is involved to generate the output.

Table 686. I2C controller configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0032h	Start bit enabled (logic 0) and stop bit enabled (logic 1).
SHIFTCTL*n*	0101_0082h	Configure transmit using timer 1 on the rising edge of clock with inverted output enable (open-drain output) on pin 0.
SHIFTCFG(*n* + 1)	0000_0020h	Start bit disabled and stop bit enabled (logic 0) for ACK or NACK detection.
SHIFTCTL(*n* + 1)	0180_0001h	Configure receive using timer 1 on the falling edge of clock with input data on pin 0.
TIMCMP*n*	0000_2501h	Configure 2 word transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of words × 18) + 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG*n*	0102_2222h	Configure start bit, stop bit, enable on trigger high, disable on compare, reset if output equals pin. Initial clock state is logic 0 and is not affected by reset.
TIMCTL*n*	01C1_0101h	Configure dual 8-bit counter using pin 1 output enable (SCL open-drain), with the shifter 0 flag as the inverted trigger.
TIMCMP(*n* + 1)	0000_000Fh	Configure 8-bit transfer. Set TIMCMP[15:0] as (number of bits x 2) - 1.
TIMCFG(*n* + 1)	0020_1112h	Enable when timer 0 is enabled; disable when timer 0 is disabled. Enable start bit and stop bit at the end of each word and decrement on pin input.
TIMCTL(*n* + 1)	01C0_0183h	Configure 16-bit counter using inverted pin 1 input (SCL).
SHIFTBUF*n*	Data to transmit	Transmit data can be written to SHIFTBUFBBS[7:0]. Use the shifter

Table continues on the next page...

Table 686. I2C controller configuration (continued)

Register	Value	Configuration
		status flag to indicate when data can be written using interrupt or DMA request.
SHIFTBUF(*n* + 1)	Data to receive	Received data can be read from SHIFTBUFBIS[7:0]. Use the shifter status flag to indicate when data can be read using interrupt or DMA request.

71.6.6 I2S controller

I2S Controller mode can be supported using two timers, two shifters, and four pins. One timer is used to generate the bit clock and control the shifters and another timer is used to generate the frame sync. FLEXIO waits for the first write to the transmit data buffer before enabling bit clock and frame sync generation. Data transfers are supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The bit clock frequency is an even integer divide of the FLEXIO clock frequency. The initial frame sync assertion occurs at the same time as the first bit clock edge. The timer uses the start bit to ensure that the frame sync is generated one clock cycle before the first output data.

NOTE

Because of synchronization delays, the setup time for the receiver input is 1.5 FLEXIO clock cycles. This means that the maximum baud rate is divide by 4 of the FLEXIO clock frequency.

Table 687. I2S controller configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0001h	Load transmit data on first shift and stop bit disabled.
SHIFTCTL*n*	0003_0002h	Configure transmit using timer 0 on the rising edge of clock with output data on pin 0.
SHIFTCFG(*n* + 1)	0000_0000h	Start and stop bit disabled.
SHIFTCTL(*n* + 1)	0080_0101h	Configure receive using timer 0 on the falling edge of clock with input data on pin 1.
TIMCMP*n*	0000_3F01h	Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG*n*	0000_0202h	Configure start bit, enable on trigger high and never disable. Initial clock state is logic 1.
TIMCTL*n*	01C3_0281h	Configure dual 8-bit counter using inverted pin 2 output (bit clock), with shifter 0 flag as the inverted trigger.

Table continues on the next page...

Table 687. I2S controller configuration (continued)

Register	Value	Configuration
		Write 0 to the PINPOL field to invert the polarity of the output shift clock.
TIMCMP(*n* + 1)	0000_007Fh	Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:0] as (number of bits × baud rate divider) ÷ 1.
TIMCFG(*n* + 1)	0000_0100h	Enable when timer 0 is enabled and never disable.
TIMCTL(*n* + 1)	0003_0383h	Configure 16-bit counter using inverted pin 3 output (as frame sync). Write 0 to the PINPOL field to invert the polarity of the output frame sync.
SHIFTBUF*n*	Data to transmit	Transmit data can be written to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) instead to support LSB first transfer.
SHIFTBUF(*n* + 1)	Data to receive	Received data can be read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) instead to support LSB first transfer.

71.6.7 I2S target

I2S Target mode can be supported using three timers, two shifters, and four pins. For single transmit and single receive, other combinations of transmit and receive are possible.

The transmit data must be written to the transmit buffer register before the external frame sync asserts, otherwise the shifter error flag is set.

NOTE

Because of synchronization delays, the output valid time for the serial output data is 2.5 FLEXIO clock cycles. This means the maximum baud rate is divide by 6 of the FLEXIO clock frequency.

The output valid time of I2S target is maximum 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization, plus one cycle to output the data.

Timer 2 detects the falling edge of frame sync (start of new frame) and asserts output until the rising edge of bit clock (when the frame sync is normally sampled). Timer 0 detects the rising edge of bit clock with timer 2 output asserted and asserts output for length of frame. Timer 1 detects the falling edge of bit clock with timer 0 output asserted and controls shift registers for 32-bit transfers.

Table 688. I2S target configuration

Register	Value	Configuration
SHIFTCFG*n*	0000_0000h	Start and stop bit disabled.
SHIFTCTL*n*	0103_0002h	Configure transmit using timer 1 on the rising edge of shift clock with output data on pin 0.
SHIFTCFG(*n* + 1)	0000_0000h	Start and stop bit disabled.
SHIFTCTL(*n* + 1)	0180_0101h	Configure receive using timer 1 on the falling edge of shift clock with input data on pin 1.
TIMCMP*n*	0000_007Fh	Configure two 32-bit transfers per frame. Set TIMCMP[15:0] as (number of bits × 4) - 1.
TIMCFG*n*	0020_2500h	Configure enable on pin rising edge (inverted bit clock) with trigger high (timer 2) and disable on compare. Initial clock state is logic 1 and decrements on pin input (bit clock).
TIMCTL*n*	0B40_0203h	Configure 16-bit counter using pin 2 input (bit clock), with timer 2 output as the trigger.
TIMCMP(*n* + 1)	0000_003Fh	Configure 32-bit transfers. Set TIMCMP[15:0] as (number of bits × 2) - 1.
TIMCFG(*n* + 1)	0020_2500h	Configure enable on pin (bit clock) rising edge with trigger (timer 0) high and disable on compare. Initial clock state is logic 1 and decrement on pin input (bit clock).
TIMCTL(*n* + 1)	0340_0283h	Configure 16-bit counter using inverted pin 2 input (bit clock), with timer 0 output as the trigger.
TIMCMP(*n* + 2)	0000_0000h	Compare on zero (first edge).
TIMCFG(*n* + 2)	0020_6400h	Configure enable on inverted pin (frame sync) rising edge and disable on trigger falling edge (bit clock). Initial clock state is logic 1 and decrement on inverted pin input (frame sync).
TIMCTL(*n* + 2)	04C0_0383h	Configure 16-bit counter using inverted pin 3 input (frame sync), with pin 2 inverted input (bit clock) as the trigger.
SHIFTBUF*n*	Data to transmit	Transmit data can be written to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be written

Table continues on the next page...

Table 688. I2S target configuration (continued)

Register	Value	Configuration
		using interrupt or DMA request. Write to the SHIFTBUF register instead to support LSB first transfer.
SHIFTBUF(*n* + 1)	Data to receive	Received data can be read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from the SHIFTBUF register instead to support LSB first transfer.

71.6.8 SENT receiver

The SENT receiver can be supported by using one timer and one pin. The timer is configured as Input-Capture mode, and captures the counter value at the falling edge of the pin input. After the counter value is captured, the counter is automatically restarted from counter value 0. Therefore, the captured value always indicates the period between the previous falling edge and current falling edge. You can configure the CPU interrupt or DMA trigger at each capture of the counter. The CPU software performs the entire SENT frame decoding with the latest tick width adjustment.

Table 689. SENT receiver configuration

Register	Value	Configuration
TIMCMP*n*	—	Stores counter value at the falling edge of the pin.
TIMCFG*n*	0000_6000h	Timer is always enabled. It is disabled on trigger falling edge, decrement counter on FLEXIO clock.
TIMCTL*n*	0040_0007h	Single 16-bit input capture mode. The PINSEL field selects the timer pin input and output. Timer pin output disabled, internal trigger selected, select pin 0 as trigger.

71.6.9 Camera interface

Camera interface can be supported using one timer, one or more shifters, and multiple pins. Multiple transfers can be supported using the DMA controller.

The example configuration shown in the following table describes the FLEXIO configuration for interfacing with an 8-bit CMOS sensor with PCLK, VSYNC, HREF, and D[7:0] outputs. The example uses a 128-bit buffer to capture 16 pixels of image data before an interrupt or DMA transfer. You can use a bigger or smaller buffer depending on system DMA performance and FLEXIO resource usage by other applications.

NOTE

You can use additional timers to track the number of pixels per row and number of rows per frame, or HREF or VSYNC can be assigned as GPIO interrupts for software tracking.

Table 690. Camera interface configuration for 8-bit CMOS sensor

Register	Value	Configuration
SHIFTCFG*n*...*n*+2[1]	0007_0100h	Configure 8-bit parallel shift in from adjacent shifter.
SHIFTCFG*n*+3	0007_0000h	Configure 8-bit parallel shift in from pins FXIO_D[7:0] (D[7:0]).
SHIFTCTL*n*...*n*+3	0080_0001h	Configure receive using timer 0 on the negative edge of clock.
TIMCMP*n*	0000_001Fh	Configure 16 pixel (8 bits/pixel × 16 pixels = 128 bits) transfer. Set TIMCMP[15:0] as (number of pixels × 2) - 1.
TIMCFG*n*	0120_6600h	Configure enable on trigger (HREF) rising edge and disable on trigger falling edge. Initial shift clock state is logic 0 and decrement on PCLK input.
TIMCTL*n*	12C0_0803h	Configure 16-bit counter using FXIO_D[8] input (PCLK), with FXIO_D[9] input (HREF) as the inverted trigger.
SHIFTBUF*n*...*n*+3	Data to receive	Received data can be read from SHIFTBUF*n*...*n*+3. Use the shifter status flag to indicate when data can be read using interrupt or DMA request.

1. *n* = 0 or 4

71.6.10 Motorola 68K and Intel 8080 bus interface

Motorola 68K and Intel 8080 bus are two parallel interfaces commonly used by smart and asynchronous LCD controllers. With GPIO, FLEXIO can drive these interfaces using one timer and one shifter. Additional shifters can be used to support large transfers via the DMA controller.

The following table provides an example of how to drive a 16-bit 68K or 8080 bus. For an 8080 bus, two GPIOs are used to drive the nCS and RS pins. For a 68K bus, an additional GPIO is required to drive the RDWR pin.

Table 691. Motorola 68K and Intel 8080 write configuration

Register	Value	Configuration
SHIFTCFG0...0	000F_0100h	Configure 16-bit parallel shift in from adjacent shifter.
SHIFTCTL0	0003_0002h	Configure transmit using timer 0 on the positive edge of clock with data output to FXIO_D[15:0].
SHIFTCTL1...0	0000_0002h	Configure transmit using timer 0 on the positive edge of clock.
TIMCMP0	0000_0101h (1 beat)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FLEXIO

Table continues on the next page...

Table 691. Motorola 68K and Intel 8080 write configuration (continued)

Register	Value	Configuration
	0000_1F01h (16 beats)	clock. Set TIMCMP[15:8] as (number of beats × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG0	0000_2200h	Configure enable on trigger high and disable on compare. Timer output high on enable.
TIMCTL0	01C3_1001h (Motorola 68K, 1 beat) 1DC3_1001h (Motorola 68K, 16 beats) 01C3_1081h (Intel 8080, 1 beat) 1DC3_1081h (Intel 8080, 16 beats)	Configure dual 8-bit counter using FXIO_D[16] output (EN pin for 68K, nWR pin for 8080), with shifter 0 (1-beat) or shifter 0 (16 beats) flag as the inverted trigger.
SHIFTBUF0...0	Data to transmit	Transmit data can be written to SHIFTBUF0 (1 beat) or SHIFTBUF0...0(16-beats) to initiate a transfer; use the shifter status flag to indicate when data can be written using interrupt or DMA request.

Table 692. Motorola 68K and Intel 8080 read configuration

Register	Value	Configuration
SHIFTCFG0...3	000F_0100h	Configure 16-bit parallel shift in from the adjacent shifter.
SHIFTCFG0	000F_0000h	Configure 16-bit parallel shift in from pin.
SHIFTCTL0...0	0080_0001h	Configure receive using timer 0 on the negative edge of clock with data input from FXIO_D[15:0].
TIMCMP0	0000_0101h (1 beat) 0000_1F01h (16 beats)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] = (number of beats × 2) - 1, and set TIMCMP[7:0] = (baud rate divider ÷ 2) - 1.
TIMCFG0	0000_2220h	Configure stop_bit. Enable on trigger high and disable on compare. Timer output high on enable.
TIMCTL0	1DC3_1001h (Motorola 68K, 1 beat) 01C3_1001h (Motorola 68K, 16 beats) 1DC3_1181h (Intel 8080, 1 beat) 01C3_1181h (Intel 8080, 16 beats)	Configure dual 8-bit counter using either FXIO_D[16] output (EN pin for 68K) or FXIO_D[17] output (nRD pin for 8080), with shifter 0 flag (1-beat) or shifter 0 flag (16 beats) as the inverted trigger.
SHIFTBUF0...0	Data received	Received data can be read from SHIFTBUF0 (1 beat) or SHIFTBUF0...0

Table continues on the next page...

Table 692. Motorola 68K and Intel 8080 read configuration (continued)

Register	Value	Configuration
		(16 beats). Use the shifter status flag to indicate when data can be read using interrupt or DMA request.

In general, any operation to a 68K or 8080 bus target begins with a register write cycle followed by one or more data read or write cycles. To accomplish this, perform the following procedure:

1. Configure FLEXIO with 1-beat write configuration.
2. Configure GPIO to assert the nCS and RS pins (and deassert the RDWR pin for 68K).
3. Write register index data to SHIFTBUF0[15:0].
4. Configure GPIO to deassert the RS pin (and assert the RDWR pin for 68K data read).
5. Configure FLEXIO with the desired read or write configuration (1 or 16 beats).
6. Use the shifter status flag to trigger interrupt or DMA driven data transfers to and from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7).
7. Configure GPIO to deassert the nCS pin.

71.6.11 Low-power state machine

Table 693 shows an example of a hypothetical state machine to illustrate the flexibility allowed in Shifter State mode.

In this example, FLEXIO waits for the FXIO_D[2] pin to assert and then drives a complementary square wave output at a frequency of FLEXIO_CLK/131072 on the FXIO_D[1:0] pins while the comparator output is asserted. This assumes that the comparator is connected to external trigger 15. See the chip-specific FLEXIO information for actual FLEXIO trigger mappings. Throughout this operation, the CPU can be kept in a stop or VLPS mode by writing 0 to CTRL[DOZEN] and ensuring that FLEXIO_CLK is enabled. Figure 434 shows the states and transitions implemented by this example.

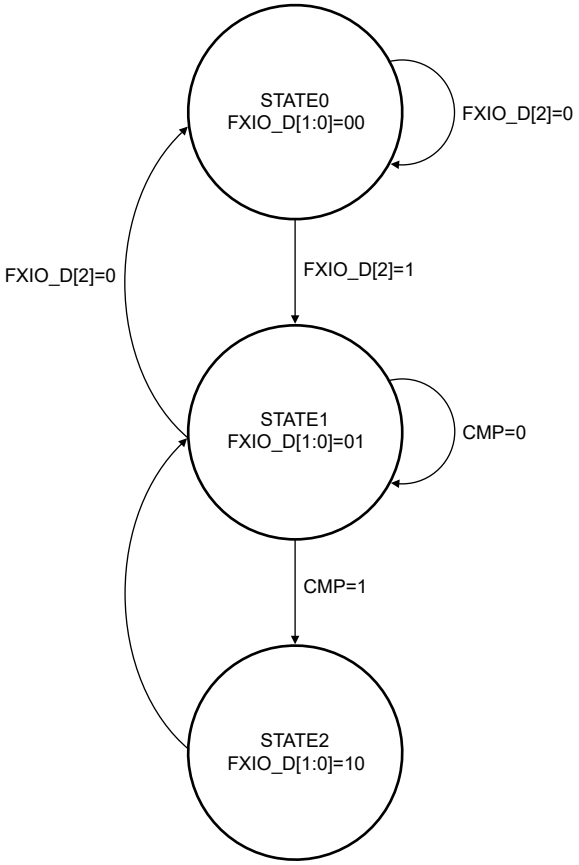


Figure 434. State diagram

Table 693. State machine configuration

Register	Value	Configuration
SHIFTCFG0...2	0000_0003h	Enable FXIO_D[1:0] as outputs.
SHIFTCTL0	0080_0206h	Configure for State mode using FXIO_D[4:2] as inputs to select next state and timer 0 output low to trigger next state.
SHIFTBUF0	0020_8208h	State0: Drive FXIO_D[1:0] = 00; transition to state0 if FXIO_D[2] = 0, state1 if FXIO_D[2] = 1.
SHIFTCTL1	0000_0206h	Configure for State mode using FXIO_D[4:2] as inputs to select next state and timer 0 output high to trigger next state.
SHIFTBUF1	0140_8408h	State1: Drive FXIO_D[1:0]=01; transition to state0 if FXIO_D[2]=0, state1 if CMP = 0, state2 if CMP = 1 (FXIO_D[3]=1).

Table continues on the next page...

Table 693. State machine configuration (continued)

Register	Value	Configuration
SHIFTCTL2	0080_0206h	Configure for State mode using FXIO_D[4:2] as inputs to select next state and timer 0 output low to trigger next state.
SHIFTBUF2	0224_9249h	State2: Drive FXIO_D[1:0] = 10, transition to state1 with timer 0 output low.
TIMCMP0	0000_FFFFh	Configure baud rate of divide by 131072 of the FLEXIO clock. Set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1.
TIMCFG0	0000_0000h	Configure timer always enabled.
TIMCTL0	0000_0003h	Configure single 16-bit counter.
TIMCFG1	0010_7600h	Configure timer enabled on trigger rising edge, disabled on trigger falling edge, decrement on trigger edge.
TIMCTL1	0F03_0303h	Configure timer output enabled on FXIO_D[3], external trigger 15 (comparator output) selected.

71.6.12 Keypad interface

The keypad interface can support a 3 × 4 keypad matrix using three timers and three shifters, although a larger matrix can be supported using additional shifters. The configuration is designed for four columns configured as active low open-drain pins and three rows configured as input pins with pull-up resistors enabled.

One shifter is configured in Logic mode to assert its output when any of the row inputs are low, indicating a key is pressed. Use a timer to filter the shifter output to ensure that the key is pressed for a minimum amount of time before performing the column scan.

A different shifter is configured for parallel transmit. Use this shifter to scan each column when a keypress is detected. When not scanning, the shifter output is configured to assert all active low open-drain column outputs to detect any keypress. Use a dedicated timer to control the transmit shifter.

The last shifter is configured for parallel receive. Use this shifter to capture the result of the column scanning so that you can decode which key (or keys) was pressed. This configuration captures the state of both row and column pins for each scan, although the row state can also be deduced by the shift order. Use a dedicated timer to control the receive shifter, which shifts at half the frequency of the transmit shifter.

When the result of the key scan is available in the receive shifter register, FLEXIO continues to monitor the row inputs and can trigger multiple scans from a single keypress. To support debouncing, you can decide how many consecutive scans must be considered as a single keypress.

NOTE

Because the pins used in Logic mode are fixed per shifter and the shifters that support parallel shifts are limited, this configuration is restricted to what pins and shifters it can use. Increasing the matrix beyond four-row inputs requires multiple shifters in Logic mode. Increasing beyond four-column outputs requires concatenating transmit shifters to create a larger shift register.

Table 694. Keypad interface configuration

Register	Value	Configuration
SHIFTCFG0	0003_0032h	Start bit enabled (logic 0) and stop bit enabled (logic 1), configured for 4-bit shift.
SHIFTCTL0	0101_0402h	Configure transmit using timer 1 on the rising edge of clock generating open-drain output on pins[7:4] (column outputs).
SHIFTBUF0	0804_0201h	Static data containing the column scan pattern; each column is scanned one-hot with dead time in between.
SHIFTCFG1	0000_0020h	In Logic mode, mask input 3.
SHIFTCTL1	0000_0007h	Configure Logic mode.
SHIFTBUF1	07FF_07FFh	Static data configuring Logic mode LUT. Output asserts if pins [3:1] are logic 0.
SHIFTCFG3	0007_0000h	Configured for 8-bit shift.
SHIFTCTL3	0280_0001h	Configure receive using timer 2 on falling edge of clock with input data on pins [7:0] (both rows and columns; pin 0 is don't care).
TIMCMP0	0000_00ffh	Configure prescanning glitch filter to 256 FLEXIO clock cycles. For different filter cycles, configure TIMCMP[15:0] as (filter cycles) - 1.
TIMCFG0	0103_6600h	Configure enable on trigger rising edge and disable on trigger falling edge. Initial clock state is logic 0 and is not affected by reset.
TIMCTL0	0540_0003h	Configure 16-bit counter using the shifter 1 flag (logic state) as trigger.
TIMCMP1	0000_0F3Fh	Configure eight shifts (twice for each column) at column scan rate of divide by 128. For a different scan frequency, define TIMCMP[7:0] as (scan divider ÷ 2) - 1.
TIMCFG1	0020_2622h	Enable on trigger rising edge, disable on compare. Start and stop bits are enabled.
TIMCTL1	0340_0001h	Configure dual 8-bit counter using timer 0 output as the trigger.
TIMCMP2	0000_0801h	Configure four shifts at half the frequency of the timer 1 trigger.

Table continues on the next page...

Table 694. Keypad interface configuration (continued)

Register	Value	Configuration
TIMCFG2	0110_2100h	Enable on timer 1 enable, disable on compare, decrement on trigger input with output initially negated, and not affected by reset.
TIMCTL2	0740_0001h	Configure dual 8-bit counter using timer 1 output as the trigger.
SHIFTBUF3	Keypad scan result	Keypad scan result can be read from SHIFTBUF3. Each byte is the result of one scan with both row and column pin state from the scan (pin 0 is not used). Use the shifter status flag to indicate when data can be written using interrupt or DMA request.

71.7 Memory map and registers

71.7.1 FLEXIO register descriptions

NOTE

Invalid register accesses, which include reading a write-only register, writing to a read-only register, or accessing an invalid address, result in a bus error.

71.7.1.1 FLEXIO memory map

FLEXIO0 base address: 4010_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0201_0003h
4h	Parameter (PARAM)	32	R	0820_0808h
8h	FLEXIO Control (CTRL)	32	RW	0000_0000h
Ch	Pin State (PIN)	32	R	0000_0000h
10h	Shifter Status (SHIFTSTAT)	32	RW	0000_0000h
14h	Shifter Error (SHIFTERR)	32	RW	0000_0000h
18h	Timer Status Flag (TIMSTAT)	32	RW	0000_0000h
20h	Shifter Status Interrupt Enable (SHIFTSIEN)	32	RW	0000_0000h
24h	Shifter Error Interrupt Enable (SHIFTEIEN)	32	RW	0000_0000h
28h	Timer Interrupt Enable (TIMIEN)	32	RW	0000_0000h
30h	Shifter Status DMA Enable (SHIFTSDEN)	32	RW	0000_0000h
38h	Timer Status DMA Enable (TIMERSDEN)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
40h	Shifter State (SHIFTSTATE)	32	RW	0000_0000h
48h	Trigger Status (TRGSTAT)	32	RW	0000_0000h
4Ch	External Trigger Interrupt Enable (TRIGIEN)	32	RW	0000_0000h
50h	Pin Status (PINSTAT)	32	RW	0000_0000h
54h	Pin Interrupt Enable (PINIEN)	32	RW	0000_0000h
58h	Pin Rising Edge Enable (PINREN)	32	RW	0000_0000h
5Ch	Pin Falling Edge Enable (PINFEN)	32	RW	0000_0000h
60h	Pin Output Data (PINOUTD)	32	RW	0000_0000h
64h	Pin Output Enable (PINOUTE)	32	RW	0000_0000h
68h	Pin Output Disable (PINOUTDIS)	32	RW	0000_0000h
6Ch	Pin Output Clear (PINOUTCLR)	32	RW	0000_0000h
70h	Pin Output Set (PINOUTSET)	32	RW	0000_0000h
74h	Pin Output Toggle (PINOUTTOG)	32	RW	0000_0000h
80h - 9Ch	Shifter Control (SHIFTCTL0 - SHIFTCTL7)	32	RW	0000_0000h
100h - 11Ch	Shifter Configuration (SHIFTCFG0 - SHIFTCFG7)	32	RW	0000_0000h
200h - 21Ch	Shifter Buffer (SHIFTBUF0 - SHIFTBUF7)	32	RW	0000_0000h
280h - 29Ch	Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7)	32	RW	0000_0000h
300h - 31Ch	Shifter Buffer Byte Swapped (SHIFTBUFBYS0 - SHIFTBUFBYS7)	32	RW	0000_0000h
380h - 39Ch	Shifter Buffer Bit Byte Swapped (SHIFTBUFBBS0 - SHIFTBUFBBS7)	32	RW	0000_0000h
400h - 41Ch	Timer Control (TIMCTL0 - TIMCTL7)	32	RW	0000_0000h
480h - 49Ch	Timer Configuration (TIMCFG0 - TIMCFG7)	32	RW	0000_0000h
500h - 51Ch	Timer Compare (TIMCMP0 - TIMCMP7)	32	RW	0000_0000h
680h - 69Ch	Shifter Buffer Nibble Byte Swapped (SHIFTBUFNBS0 - SHIFTBUFNBS7)	32	RW	0000_0000h
700h - 71Ch	Shifter Buffer Halfword Swapped (SHIFTBUFHWS0 - SHIFTBUFHWS7)	32	RW	0000_0000h
780h - 79Ch	Shifter Buffer Nibble Swapped (SHIFTBUFNIS0 - SHIFTBUFNIS7)	32	RW	0000_0000h
800h - 81Ch	Shifter Buffer Odd Even Swapped (SHIFTBUFOES0 - SHIFTBUFOES7)	32	RW	0000_0000h
880h - 89Ch	Shifter Buffer Even Odd Swapped (SHIFTBUFEOS0 - SHIFTBUFEOS7)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
900h - 91Ch	Shifter Buffer Halfword Byte Swapped (SHIFTBUFHBS0 - SHIFTBUFHBS7)	32	RW	0000_0000h

71.7.1.2 Version ID (VERID)

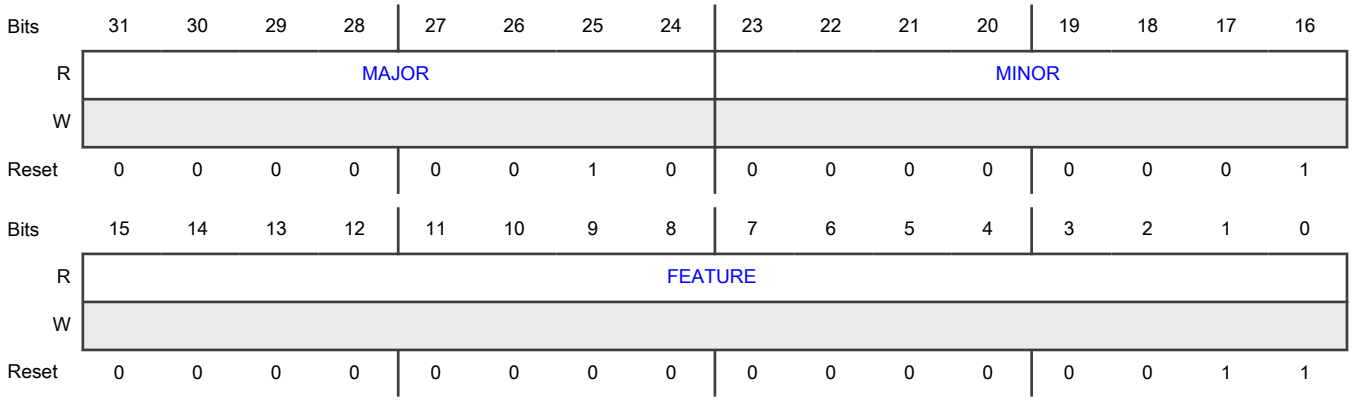
Offset

Register	Offset
VERID	0h

Function

Indicates the version of FLEXIO.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Indicates the major version number of the module specification.
23-16 MINOR	Minor Version Number Indicates the minor version number of the module specification.
15-0 FEATURE	Feature Specification Number Indicates the feature set number. 0000_0000_0000_0000b - Standard features implemented

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000_0000_0000_0001b - State, logic, and parallel modes supported 0000_0000_0000_0010b - Pin control registers supported 0000_0000_0000_0011b - State, logic, and parallel modes, plus pin control registers supported

71.7.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function

Contains the number of shifters, timers, pins, and triggers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TRIGGER								PIN							
W																
Reset	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMER								SHIFTER							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Fields

Field	Function
31-24 TRIGGER	Trigger Number Indicates the number of external triggers implemented.
23-16 PIN	Pin Number Indicates the number of pins implemented.
15-8 TIMER	Timer Number Indicates the number of timers implemented.
7-0 SHIFTER	Shifter Number Indicates the number of shifters implemented.

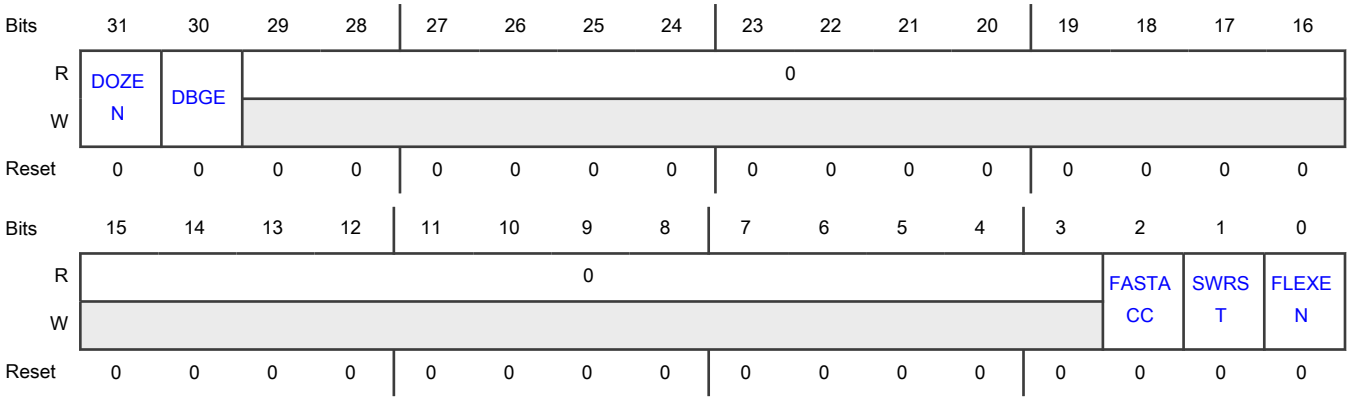
71.7.1.4 FLEXIO Control (CTRL)

Offset

Register	Offset
CTRL	8h

Function
Controls various aspects of the FLEXIO operation.

Diagram



Fields

Field	Function
31 DOZEN	Doze Enable Disables FLEXIO operation in Doze modes. 0b - Enable 1b - Disable
30 DBGE	Debug Enable Enables the FLEXIO operation in Debug mode. 0b - Disable 1b - Enable
29-3 —	Reserved
2 FASTACC	Fast Access Configures fast or normal register accesses to FLEXIO registers, but requires the FLEXIO functional clock to be at least equal to the frequency of the bus clock.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Normal 1b - Fast
1 SWRST	Software Reset Specifies whether software reset is enabled. The software reset does not affect this register but it affects all other logic in FLEXIO. All other register accesses are ignored until this field is cleared. The field remains 1 until software clears it and the reset has cleared in the FLEXIO clock domain. If you write 1 to this field, all FLEXIO registers except the Control register are reset. 0b - Disabled 1b - Enabled
0 FLEXEN	FLEXIO Enable Enables FLEXIO. 0b - Disable 1b - Enable

71.7.1.5 Pin State (PIN)

Offset

Register	Offset
PIN	Ch

Function
Indicates the status of the pin data input.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PDI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PDI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PDI	Pin Data Input Indicates the input data on each of the FLEXIO pins.

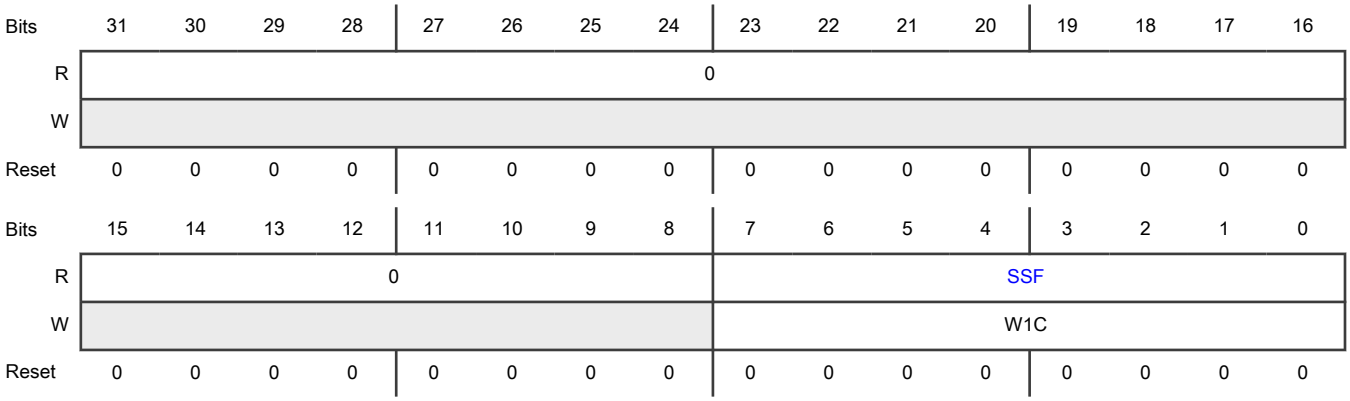
71.7.1.6 Shifter Status (SHIFTSTAT)

Offset

Register	Offset
SHIFTSTAT	10h

Function
Contains shifter status flags.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SSF	Shifter Status Flag Indicates the shifter status. This flag is updated in one of the following cases: • If SHIFTCTL*n*[SMOD] = 001b (Receive mode), the status flag is set when SHIFTBUF is loaded with data from the shifter (SHIFTBUF is full). The status flag is cleared when you read Shifter Buffer (SHIFTBUF0 - SHIFTBUF7).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• If SHIFTCTL*n*[SMOD] = 010b (Transmit mode), the status flag is set when SHIFTBUF data is transferred to the shifter (SHIFTBUF is empty) or when SHIFTCTL*n*[SMOD] is initially configured as 010b (Transmit mode). The status flag is cleared when you write to the SHIFTBUF register. • If SHIFTCTL*n*[SMOD] = 100b (Match Store mode), the status flag is set when a match occurs between SHIFTBUF and the shifter. The status flag is cleared when you read the SHIFTBUF register. • If SHIFTCTL*n*[SMOD] = 101b (Match Continuous mode), the status flag returns the current match result between SHIFTBUF and the shifter. You cannot clear the status flag by reading the SHIFTBUF register. • If SHIFTCTL*n*[SMOD] = 110b (State mode), the status flag for a shifter sets when it is selected by the current state pointer. • If SHIFTCTL*n*[SMOD] = 111b (Logic mode), the status flag returns the current value of the programmable logic block output. You can clear this status flag by writing a logic one to the flag for all modes except Match Continuous mode, State mode, and Logic mode. **NOTE** This field behaves differently for register reads and writes. When reading 0000_0000b - Clear 0000_0001b - Set When writing 0000_0000b - No effect 0000_0001b - Clear the flag

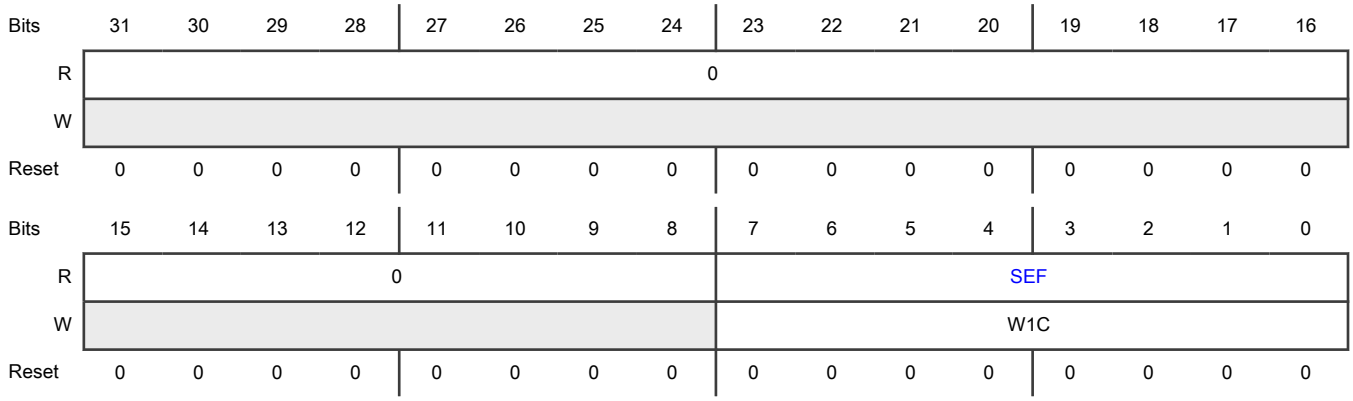
71.7.1.7 Shifter Error (SHIFTERR)

Offset

Register	Offset
SHIFTERR	14h

Function
Reports shifter errors.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SEF	Shifter Error Flag Indicates shifter error flag status. This flag is set when one of the following events occurs: • If SHIFTCTL*n*[SMOD] = 001b (Receive mode), it indicates that either the shifter is ready to store new data into SHIFTBUF before the previous data is read from SHIFTBUF (SHIFTBUF overrun), or the received start or stop bit does not match the expected value. • If SHIFTCTL*n*[SMOD] = 010b (Transmit mode), it indicates that the shifter is ready to load new data from SHIFTBUF before new data is written into SHIFTBUF (SHIFTBUF underrun). • If SHIFTCTL*n*[SMOD] = 100b (Match Store mode), it indicates the occurrence of a match event before the previous match data is read from SHIFTBUF (SHIFTBUF overrun). • If SHIFTCTL*n*[SMOD] = 101b (Match Continuous mode), the error flag is set when a match occurs between SHIFTBUF and the shifter. • If SHIFTCTL*n*[SMOD] = 111b (Logic mode), the error flag is set when the output of the programmable logic block is asserted. For SHIFTCTL*n*[SMOD] = 101b (Match Continuous mode), the flag can also be cleared when you read Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). **NOTE** This field behaves differently for register reads and writes. When reading 0000_0000b - Clear 0000_0001b - Set When writing 0000_0000b - No effect 0000_0001b - Clear the flag

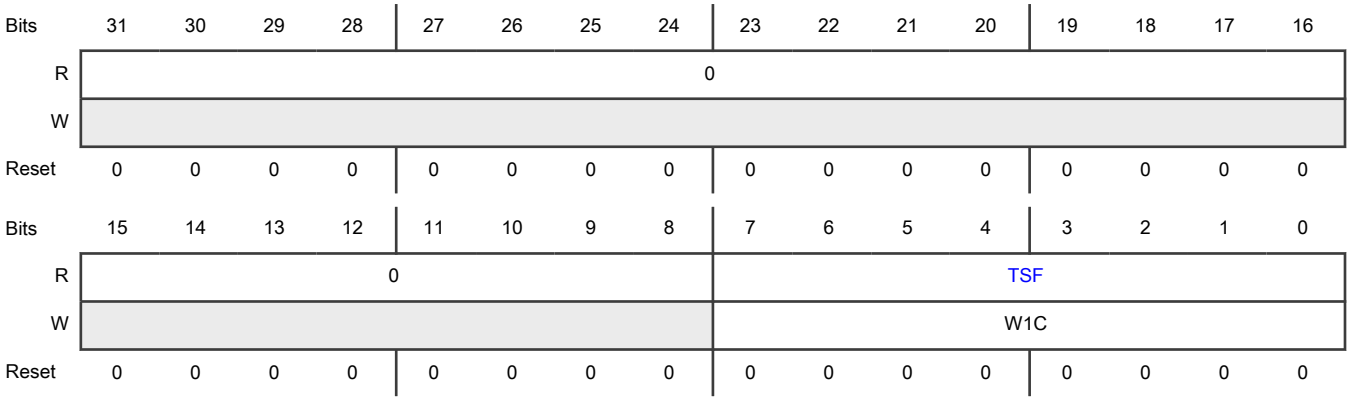
71.7.1.8 Timer Status Flag (TIMSTAT)

Offset

Register	Offset
TIMSTAT	18h

Function
Reports timer status.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 TSF	Timer Status Flag Indicates timer status. This flag is set depending on Timer mode: • In 8-bit baud counter mode, this flag is set when the upper 8-bit counter equals zero and decrements. • In 8-bit high PWM mode, this flag is set when the upper 8-bit counter equals zero and decrements. • In 16-bit counter mode, this flag is set when the 16-bit counter equals zero and decrements. • In 16-bit counter disable mode, TSF is set when a timer disable event is detected. • In 8-bit word counter mode, TSF is set when the upper 8-bit counter equals zero and decrements. • In 8-bit low PWM mode, TSF is set when the upper 8-bit counter equals zero and decrements. • In 16-bit input capture mode, TSF is set when a timer disable event is detected and the flag is clear. In this mode, you must read Timer Control (TIMCTL0 - TIMCTL7) only when TSF is set. **NOTE** This field behaves differently for register reads and writes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When reading 0000_0000b - Clear 0000_0001b - Set When writing 0000_0000b - No effect 0000_0001b - Clear the flag

71.7.1.9 Shifter Status Interrupt Enable (SHIFTSIEN)

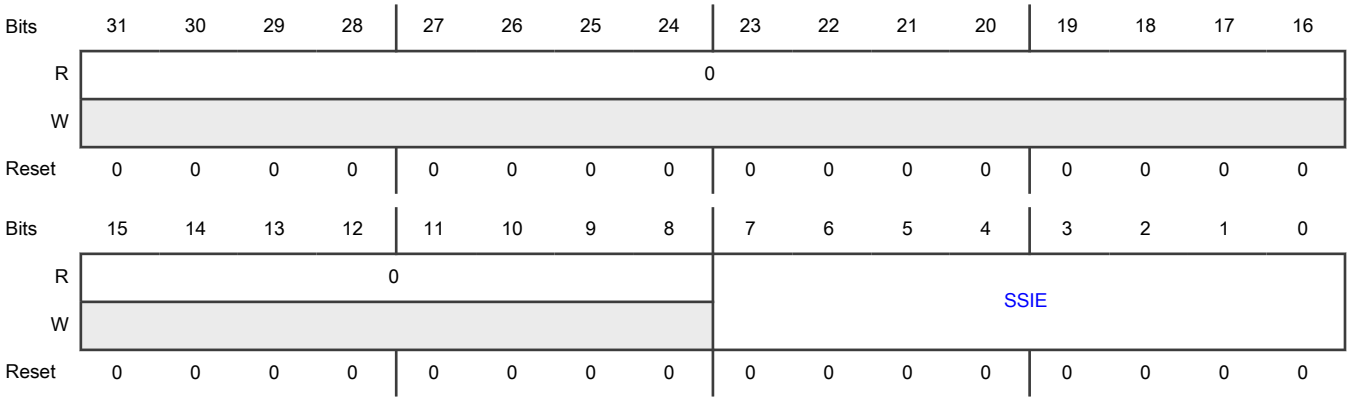
Offset

Register	Offset
SHIFTSIEN	20h

Function

Enables shifter status interrupts.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SSIE	Shifter Status Interrupt Enable Enables interrupt generation when the corresponding SHIFTSTAT[SSF] flag is set. If you write 0 to this field, SHIFTSTAT[SSF] is disabled; and if you write 1 to this field, SHIFTSTAT[SSF] is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable

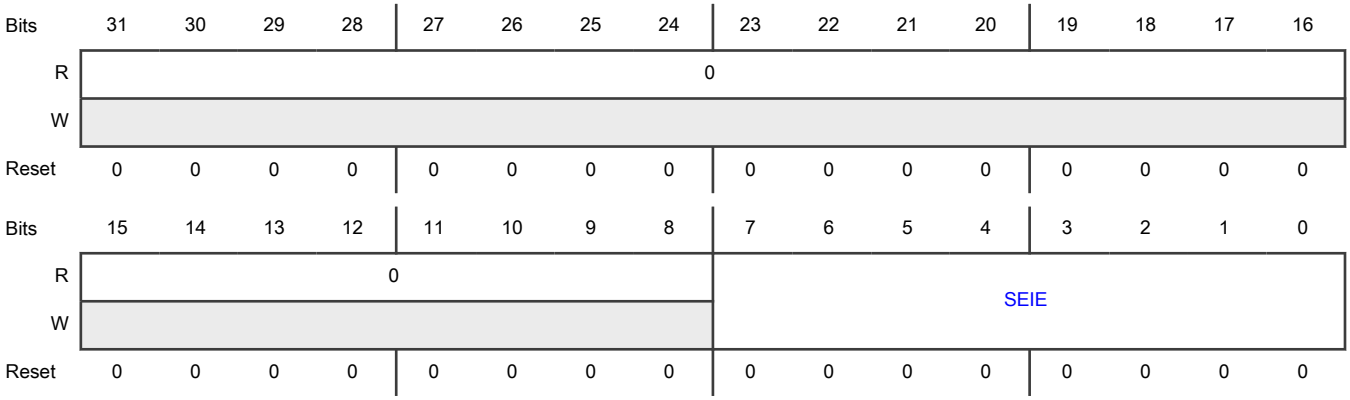
71.7.1.10 Shifter Error Interrupt Enable (SHIFTEIEN)

Offset

Register	Offset
SHIFTEIEN	24h

Function
Enables shifter error interrupts.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SEIE	Shifter Error Interrupt Enable Enables interrupt generation when the corresponding SHIFTERR[SEF] flag is set. If you write 0 to this field, SHIFTERR[SEF] is disabled; and if you write 1 to this field, SHIFTERR[SEF] is enabled. 0b - Disable 1b - Enable

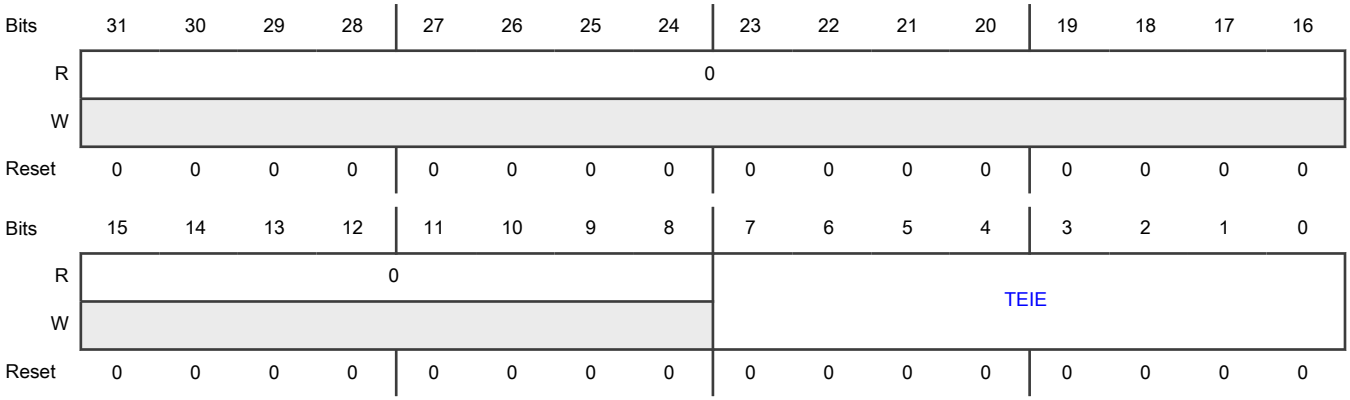
71.7.1.11 Timer Interrupt Enable (TIMIEN)

Offset

Register	Offset
TIMIEN	28h

Function
Enables timer status interrupts.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 TEIE	Timer Status Interrupt Enable Enables interrupt generation when the corresponding TIMSTAT[TSF] flag is set. If you write 0 to this field, TIMSTAT[TSF] is disabled; and if you write 1 to this field, TIMSTAT[TSF] is enabled. 0b - Disable 1b - Enable

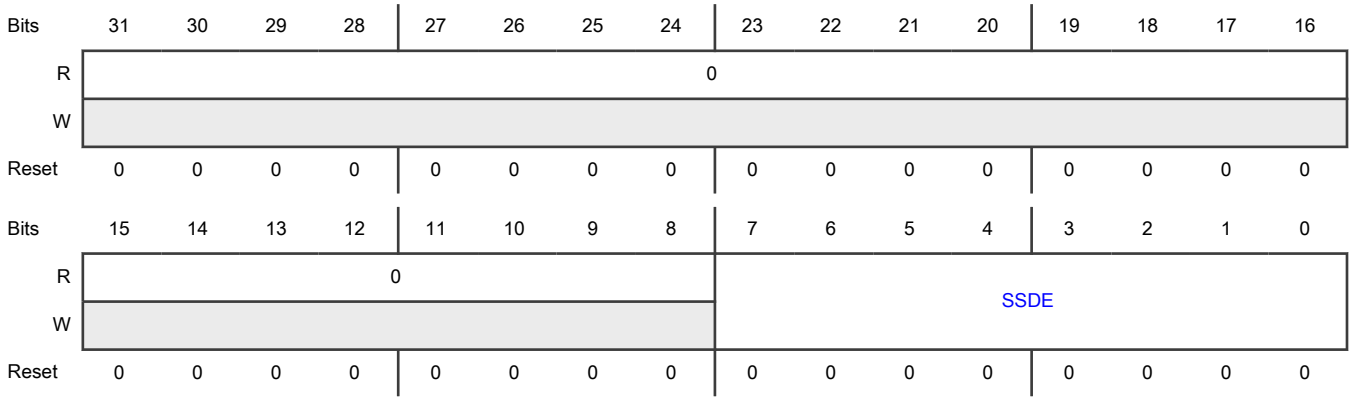
71.7.1.12 Shifter Status DMA Enable (SHIFTSDEN)

Offset

Register	Offset
SHIFTSDEN	30h

Function
Enables shifter DMA requests.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SSDE	Shifter Status DMA Enable Enables DMA request generation when the corresponding SHIFTSTAT[SSF] flag is set. If you write 0 to this field, SHIFTSTAT[SSF] is disabled; and if you write 1 to this field, SHIFTSTAT[SSF] is enabled. 0b - Disable 1b - Enable

71.7.1.13 Timer Status DMA Enable (TIMERSDEN)

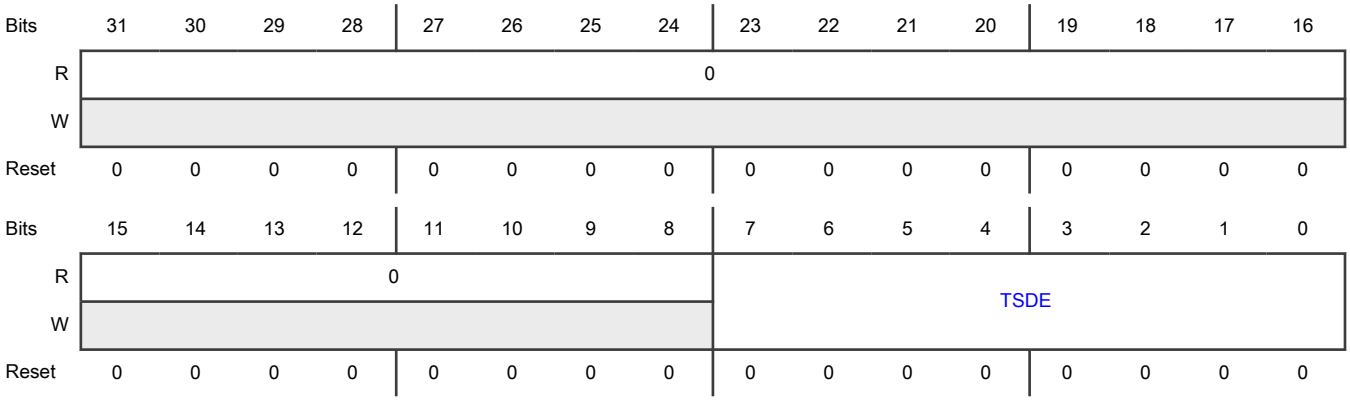
Offset

Register	Offset
TIMERSDEN	38h

Function

Enables DMA requests when the timer status flag is set.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 TSDE	Timer Status DMA Enable Enables DMA request generation when the corresponding TIMSTAT[TSF] flag is set. When the timer status DMA request is enabled, reading or writing to a timer compare register clears the corresponding timer status register. The DMA must therefore read or write to the timer compare register as part of the DMA transfer; otherwise, the DMA request remains asserted. 0b - Disable 1b - Enable

71.7.1.14 Shifter State (SHIFTSTATE)

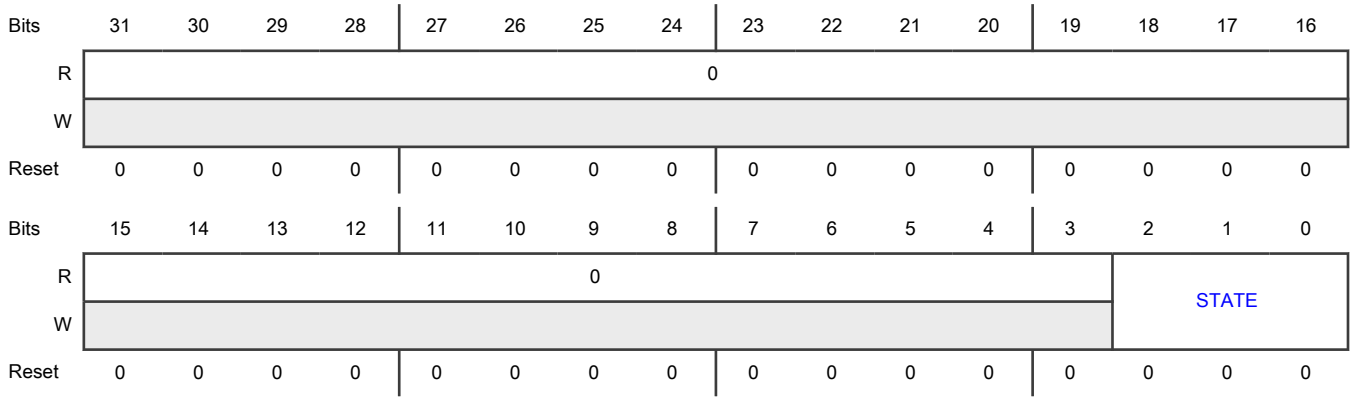
Offset

Register	Offset
SHIFTSTATE	40h

Function

Contains a pointer to track the current shifter.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 STATE	Current State Pointer Maintains a pointer to track the current shifter (configured for State mode) enabled to drive outputs and compute the next state. Reading this register when the state pointer is updating can result in the return of an incorrect state. The value that you write to this field overrides the current state.

71.7.1.15 Trigger Status (TRGSTAT)

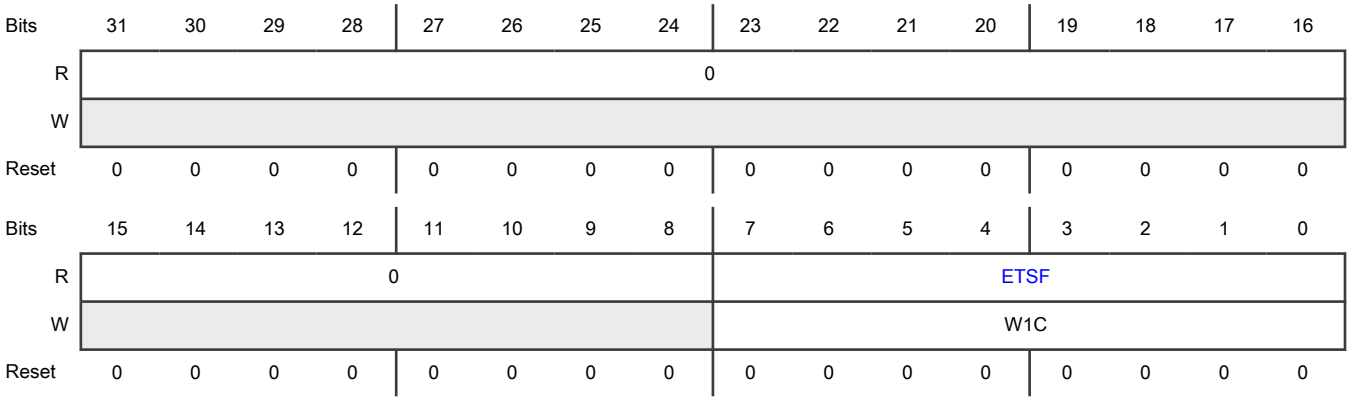
Offset

Register	Offset
TRGSTAT	48h

Function

Contains external trigger status flags.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 ETSF	External Trigger Status Flag Specifies whether the external trigger status flag is set when a rising edge is detected on the corresponding external trigger input. **NOTE** This field behaves differently for register reads and writes. When reading 0000_0000b - Clear 0000_0001b - Set When writing 0000_0000b - No effect 0000_0001b - Clear the flag

71.7.1.16 External Trigger Interrupt Enable (TRIGIEN)

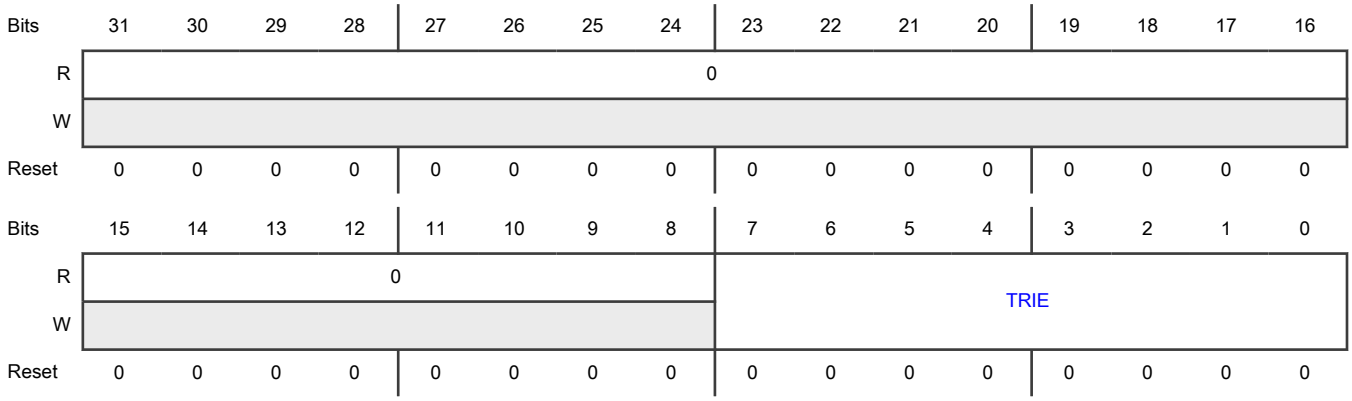
Offset

Register	Offset
TRIGIEN	4Ch

Function

Enables external trigger interrupts.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 TRIE	External Trigger Interrupt Enable Enables interrupt generation when the corresponding TRGSTAT[ETSF] flag is set. If you write 0 to this field, TRGSTAT[ETSF] is disabled, and if you write 1 to this field, TRGSTAT[ETSF] is enabled. 0b - Disable 1b - Enable

71.7.1.17 Pin Status (PINSTAT)

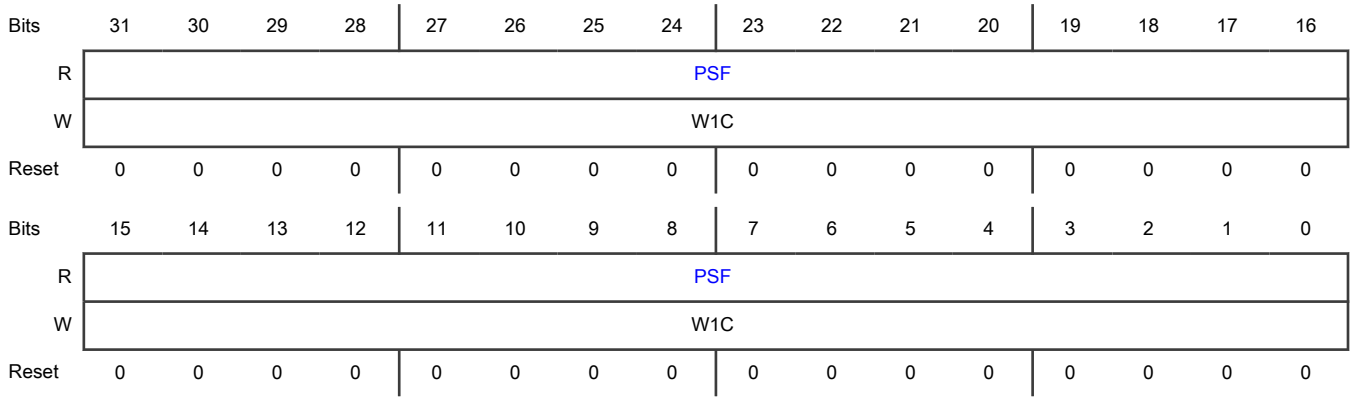
Offset

Register	Offset
PINSTAT	50h

Function

Contains pin status flags.

Diagram



Fields

Field	Function
31-0 PSF	Pin Status Flag Indicates whether the pin status flag is set when a rising edge or falling edge (if configured) is detected on the corresponding pin, as configured by the pin. **NOTE** This field behaves differently for register reads and writes. When reading 0000_0000_0000_0000_0000_0000_0000_0000b - Clear 0000_0000_0000_0000_0000_0000_0000_0001b - Set When writing 0000_0000_0000_0000_0000_0000_0000_0000b - No effect 0000_0000_0000_0000_0000_0000_0000_0001b - Clear the flag

71.7.1.18 Pin Interrupt Enable (PINIEN)

Offset

Register	Offset
PINIEN	54h

Function

Enables pin status interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PSIE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PSIE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PSIE	Pin Status Interrupt Enable Enables interrupt generation when the corresponding PINSTAT[PSF] flag is set. If you write 0 to this field, PINSTAT[PSF] is disabled, and if you write 1 to this field, PINSTAT[PSF] is enabled. 0b - Disable 1b - Enable

71.7.1.19 Pin Rising Edge Enable (PINREN)

Offset

Register	Offset
PINREN	58h

Function
Enables the pin status flag on a rising edge.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PRE	Pin Rising Edge Specifies whether the pin status flag is set whenever a rising edge is detected on the pin. 0b - Not set 1b - Set

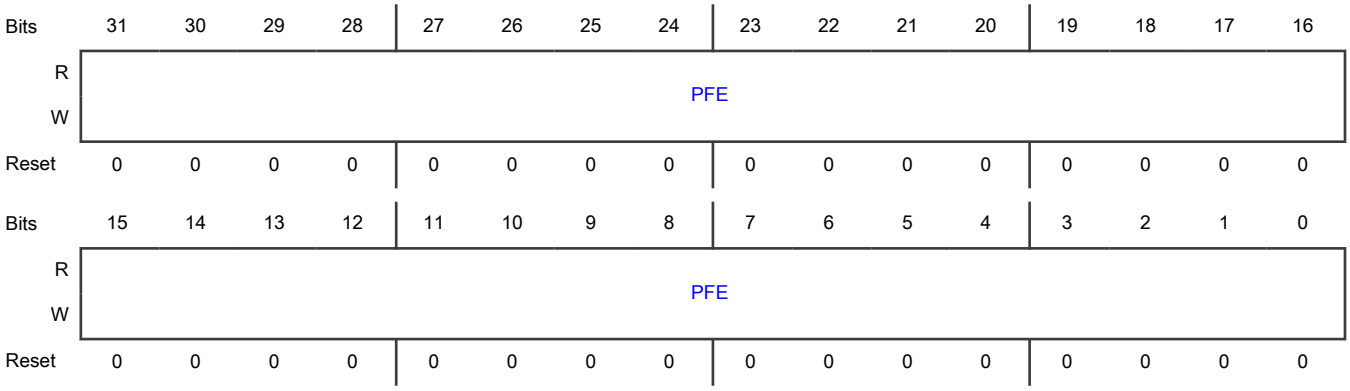
71.7.1.20 Pin Falling Edge Enable (PINFEN)

Offset

Register	Offset
PINFEN	5Ch

Function
Enables the pin status flag on a falling edge.

Diagram



Fields

Field	Function
31-0 PFE	Pin Falling Edge Specifies whether the pin status flag is set whenever a falling edge is detected on the pin. 0b - Not set 1b - Set

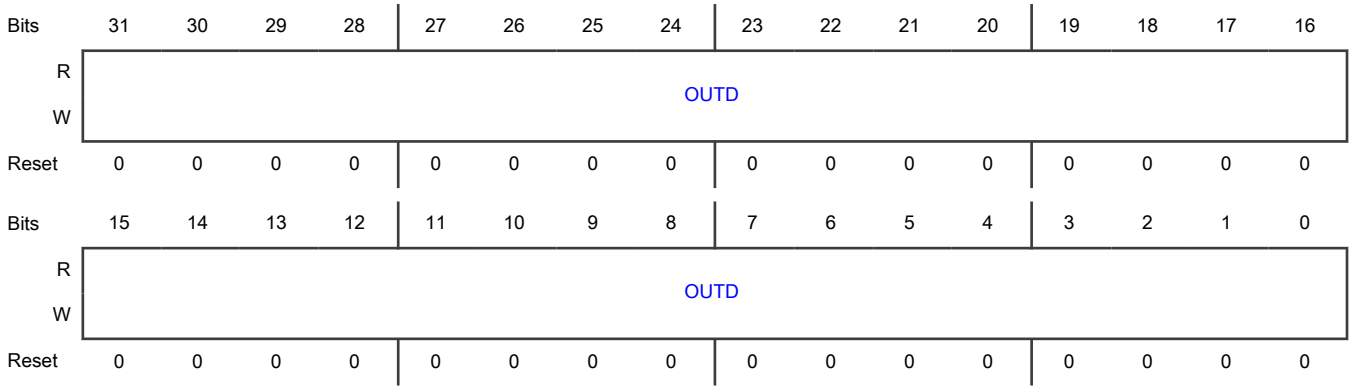
71.7.1.21 Pin Output Data (PINOUTD)

Offset

Register	Offset
PINOUTD	60h

Function
Contains data output when direct pin output is enabled.

Diagram



Fields

Field	Function
31-0 OUTD	Output Data Configures the value driven on the corresponding pin when direct pin output is enabled. 0b - Logic zero 1b - Logic one

71.7.1.22 Pin Output Enable (PINOUTE)

Offset

Register	Offset
PINOUTE	64h

Function
Enables pin output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OUTE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUTE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 OUTE	Output Enable Enables direct output on the corresponding pin. If this field is 0, the pin is controlled by timer/shifter configuration, and if this field is 1, pin is an output and driven with the value of Pin Output Data (PINOUTD). 0b - Controlled by timer/shifter configuration 1b - Output; driven with value of PINOUTD

71.7.1.23 Pin Output Disable (PINOUTDIS)

Offset

Register	Offset
PINOUTDIS	68h

Function
Disables pin output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	OUTDIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	OUTDIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 OUTDIS	Output Disable Configures the corresponding pins to disable direct output. If this field is 1, the corresponding fields in Pin Output Data (PINOUTD) and Pin Output Enable (PINOUTE) become 0. 0b - No effect 1b - Corresponding fields become 0

71.7.1.24 Pin Output Clear (PINOUTCLR)

Offset

Register	Offset
PINOUTCLR	6Ch

Function
Clears pin output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	OUTCLR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	OUTCLR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 OUTCLR	Output Clear Configures the corresponding pins to output zero. If this field is 1, the corresponding field in Pin Output Data (PINOUTD) becomes 0 and the one in Pin Output Enable (PINOUTE) becomes 1. 0b - No effect 1b - Corresponding field in PINOUTD becomes 0; corresponding field in PINOUTE becomes 1

71.7.1.25 Pin Output Set (PINOUTSET)

Offset

Register	Offset
PINOUTSET	70h

Function
Sets pin output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	OUTSET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	OUTSET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 OUTSET	Output Set Configures the corresponding pins to output logic one. If this field is 1, the corresponding fields in Pin Output Data (PINOUTD) and Pin Output Enable (PINOUTE) become 1. 0b - No effect 1b - Corresponding fields become 1

71.7.1.26 Pin Output Toggle (PINOUTTOG)

Offset

Register	Offset
PINOUTTOG	74h

Function
Toggles pin output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	OUTTOG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	OUTTOG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 OUTTOG	Output Toggle Configures the corresponding pins to toggle. If this field is 1, the corresponding field in Pin Output Data (PINOUTD) is inverted and the one in Pin Output Enable (PINOUTE) becomes 1. 0b - No effect 1b - Corresponding field in PINOUTD is inverted; corresponding field in PINOUTE becomes 1

71.7.1.27 Shifter Control (SHIFTCTL0 - SHIFTCTL7)

Offset

For n = 0 to 7:

Register	Offset
SHIFTCTLn	80h + (n × 4h)

Function

Provides shifter controls.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				TIMSEL				TIMPOL	0			PINCFG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			PINSEL					PINPOL	0			SMOD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-27 —	Reserved
26-24 TIMSEL	Timer Select Selects which timer is used for controlling the logic or shift register and generating the shift clock. TIMSEL = i selects TIMERi.
23 TIMPOL	Timer Polarity Determines whether the shift occurs on the positive edge or negative edge of the shift clock. 0b - Positive edge 1b - Negative edge
22-18 —	Reserved
17-16 PINCFG	Shifter Pin Configuration Specifies shifter pin configuration. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register. **NOTE** When initially configuring PINCFG as 11b, FLEXIO may briefly drive the pin low. To avoid this, you can configure PINCFG as 10b along with the rest of the Control register and then perform a subsequent write to set PINCFG as 11b. Likewise, when changing the value of PINCFG from 11b to 00b, you must perform an initial write to set PINCFG as 10b and then perform a subsequent write to update the rest of the Control register with the value of PINCFG as 00b. 00b - Shifter pin output disabled 01b - Shifter pin open-drain or bidirectional output enable 10b - Shifter pin bidirectional output data 11b - Shifter pin output
15-13 —	Reserved
12-8 PINSEL	Shifter Pin Select Selects the pin that is used by the shifter input or output. PINSEL = i selects the FXIO_Di pin. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register.
7 PINPOL	Shifter Pin Polarity Specifies the shifter pin polarity. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to this register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Active high 1b - Active low
6-3 —	Reserved
2-0 SMOD	Shifter Mode Configures the mode of the shifter. 000b - Disable 001b - Receive mode; capture the current shifter content into SHIFTBUF on expiration of the timer 010b - Transmit mode; load SHIFTBUF contents into the shifter on expiration of the timer 011b - Reserved 100b - Match Store mode; shifter data is compared to SHIFTBUF content on expiration of the timer 101b - Match Continuous mode; shifter data is continuously compared to SHIFTBUF contents 110b - State mode; SHIFTBUF contents store programmable state attributes 111b - Logic mode; SHIFTBUF contents implement programmable logic lookup table

71.7.1.28 Shifter Configuration (SHIFTCFG0 - SHIFTCFG7)

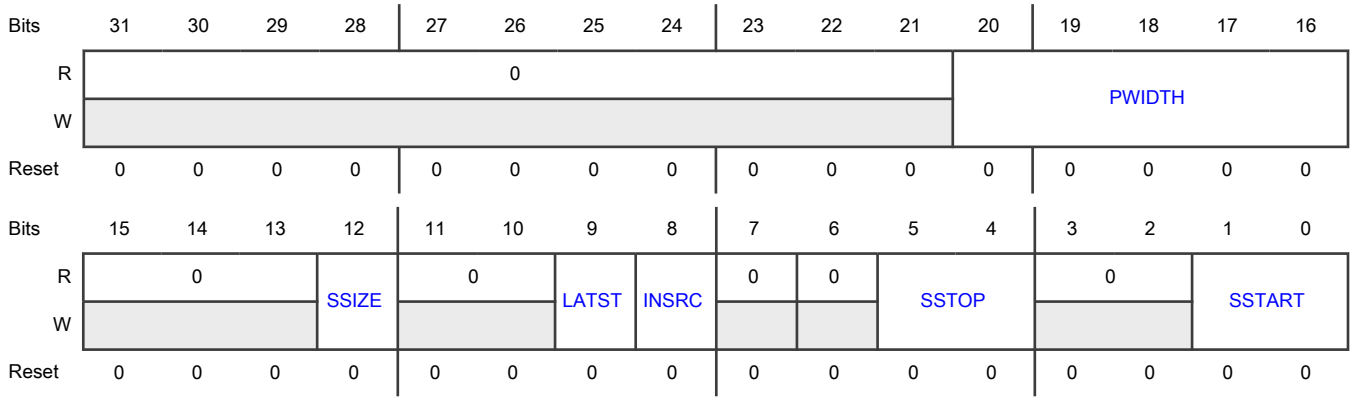
Offset

For n = 0 to 7:

Register	Offset
SHIFTCFGn	100h + (n × 4h)

Function
Provides fields for shifter configuration.

Diagram



Fields

Field	Function
31-21 —	Reserved
20-16 PWIDTH	Parallel Width Configures the number of bits to be shifted on each shift clock for all shifters: • 1-bit shift for PWIDTH = 0 • 2-bit shift for PWIDTH = 1 • 4-bit shift for PWIDTH = 2...3 • 8-bit shift for PWIDTH = 4...7 • 16-bit shift for PWIDTH = 8...15 • 32-bit shift for PWIDTH = 16...31 For shifters that support parallel transmit (SHIFTER0, SHIFTER4, ...) or parallel receive (SHIFTER3, SHIFTER7, ...), this field, together with PINSEL, also selects the pins to be driven or sampled on each shift clock: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL]. Shifters that do not support parallel transmit or parallel receive only support parallel shift when SHIFTCFG*n*[INSRC] = 1. If SHIFTCTL*n*[SMOD] = 110b (State mode), use this field to disable state outputs (see State mode).
15-13 —	Reserved
12 SSIZE	Shifter Size Configures the size of the Shift registers. A 24-bit Shift register shifts data only into bits [23:0] and does not update bits [31:24] during shift operations. When the Shift register is configured for a 24-bit shift, configuring PWIDTH as 8..15 performs a 12-bit shift and PWIDTH as 16..31 performs a 24-bit shift.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - 32-bit 1b - 24-bit
11-10 —	Reserved
9 LATST	Late Store Configures what happens when a receive or match Shift register is configured to both shift and store on the same cycle. 0b - Store the pre-shift register state 1b - Store the post-shift register state
8 INSRC	Input Source Selects the input source for the shifter. Configuring this field as 1 is not supported for the last shifter. 0b - Pin 1b - Shifter n+1 output
7 —	Reserved
6 —	Reserved
5-4 SSTOP	Shifter Stop Allows automatic stop bit insertion, if the selected timer has also enabled a stop bit, when SHIFTCTL*n*[SMOD] is 10b (Transmit mode). If SHIFTCTL*n*[SMOD] is 1b or 100b (Receive mode or Match Store mode), this field allows automatic stop bit checking if the selected timer has also enabled a stop bit. If SHIFTCTL*n*[SMOD] is 110b (State mode), this field disables state outputs (see State mode). If SHIFTCTL*n*[SMOD] is 111b (Logic mode), this field masks logic pin inputs (see Logic mode). 00b - Stop bit disabled for Transmitter, Receiver, and Match Store modes 01b - Stop bit disabled for Transmitter, Receiver, and Match Store modes; when timer is in stop condition, Receiver and Match Store modes store receive data on the configured shift edge 10b - Transmitter mode outputs stop bit value 0 in Match Store mode; if stop bit is not 0, Receiver and Match Store modes set error flag (when timer is in stop condition, these modes also store receive data on the configured shift edge) 11b - Transmitter mode outputs stop bit value 1 in Match Store mode; if stop bit is not 1, Receiver and Match Store modes set error flag (when timer is in stop condition, these modes also store receive data on the configured shift edge)
3-2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1-0 SSTART	Shifter Start Allows automatic start bit insertion, if the selected timer has also enabled a start bit, when SHIFTCTL*n*[SMOD] is 10b (Transmit mode). If SHIFTCTL*n*[SMOD] = 1b (Receive mode) or 100b (Match Store mode), this field allows automatic start bit checking if the selected timer has also enabled a start bit. If SHIFTCTL*n*[SMOD] is 110b (State mode), this field disables state outputs (see State mode). If SHIFTCTL*n*[SMOD] = 111b (Logic mode), this field masks logic pin inputs (see Logic mode). 00b - Start bit disabled for Transmitter, Receiver, and Match Store modes; Transmitter mode loads data on enable 01b - Start bit disabled for Transmitter, Receiver, and Match Store modes; Transmitter mode loads data on first shift 10b - Transmitter mode outputs start bit value 0 before loading data on first shift; if start bit is not 0, Receiver and Match Store modes set error flag 11b - Transmitter mode outputs start bit value 1 before loading data on first shift; if start bit is not 1, Receiver and Match Store modes set error flag

71.7.1.29 Shifter Buffer (SHIFTBUF0 - SHIFTBUF7)

Offset

For n = 0 to 7:

Register	Offset
SHIFTBUFn	200h + (n × 4h)

Function

Contains shift buffer data.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SHIFTBUF	Shift Buffer Contains the data to be matched with the shifter contents and is used for various other functions, depending on the setting of SHIFTCTL*n*[SMOD]: • If SHIFTCTL0[SMOD] is 1b (Receive mode), shifter data is transferred into SHIFTBUF at the expiration of the timer. You must read this register only when the corresponding SHIFTSTAT[SSF] flag is set, indicating that new shifter data is available. • If SHIFTCTL0[SMOD] is 10b (Transmit mode), SHIFTBUF data is transferred into the shifter before the timer begins. • If SHIFTCTL0[SMOD] is 100b (Match Store mode), SHIFTBUF[31:16] contains the data to be matched with the shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1 = mask, 0 = no mask). The match is checked when the timer expires. Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs. You must read this register only when the corresponding shifter status flag is set, indicating that new shifter data is available. • If SHIFTCTL0[SMOD] is 101b (Match Continuous mode), SHIFTBUF[31:16] contains the data to be matched with the shifter contents, and SHIFTBUF[15:0] can be used to mask the match result (1 = mask, 0 = no mask). • If SHIFTCTL0[SMOD] is 111b (Logic mode), SHIFTBUF[31:0] implements a 5-input, 32-bit programmable logic lookup table (see Logic mode). • If SHIFTCTL0[SMOD] is 110b (State mode), use SHIFTBUF[31:24] to drive the output value when this shifter is selected by the current state pointer and use SHIFTBUF[23:0] to configure the value of the next state transition (see State mode).

71.7.1.30 Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7)

Offset

For n = 0 to 7:

Register	Offset
SHIFTBUFBISn	280h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is bit-swapped.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFBIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFBIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SHIFTBUFBIS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are bit-swapped. Reads return SHIFTBUF[0:31].

71.7.1.31 Shifter Buffer Byte Swapped (SHIFTBUFBYS0 - SHIFTBUFBYS7)

Offset

For n = 0 to 7:

Register	Offset
SHIFTBUFBYSn	300h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is byte-swapped.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFBYS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFBYS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SHIFTBUFBYS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are byte-swapped. Reads return {SHIFTBUF[7:0], SHIFTBUF[15:8], SHIFTBUF[23:16], SHIFTBUF[31:24]}.

71.7.1.32 Shifter Buffer Bit Byte Swapped (SHIFTBUFBBS0 - SHIFTBUFBBS7)

Offset

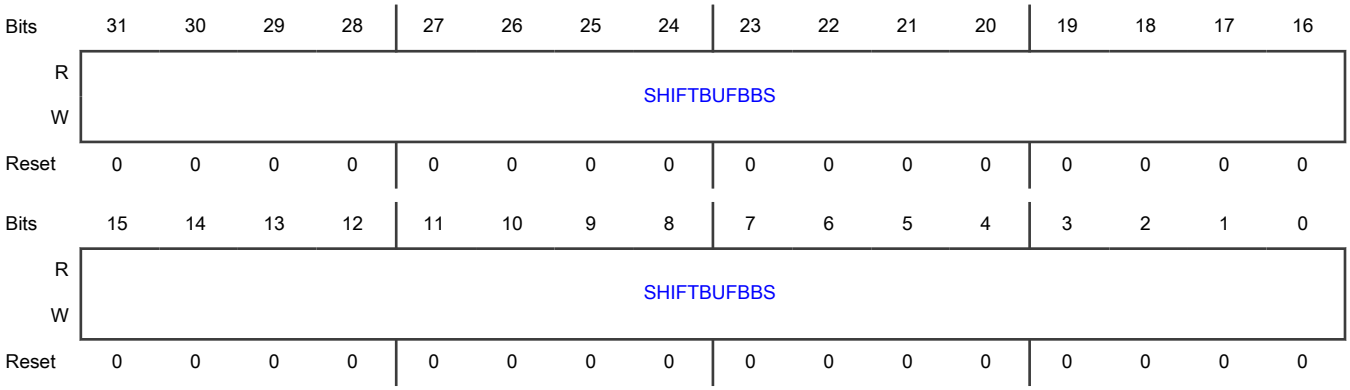
For n = 0 to 7:

Register	Offset
SHIFTBUFBBSn	380h + (n × 4h)

Function

Contains the register data for Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but it is bit-swapped within each byte.

Diagram



Fields

Field	Function
31-0 SHIFTBUFBBS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), except that reads or writes to this register are bit-swapped within each byte. Reads return {SHIFTBUF[24:31], SHIFTBUF[16:23], SHIFTBUF[8:15], SHIFTBUF[0:7]}.

71.7.1.33 Timer Control (TIMCTL0 - TIMCTL7)

Offset

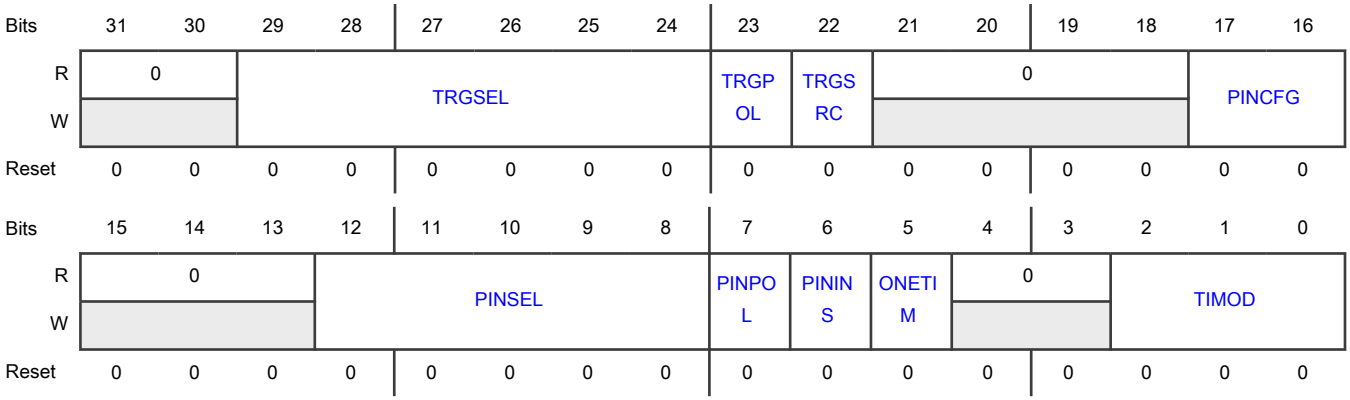
For n = 0 to 7:

Register	Offset
TIMCTLn	400h + (n × 4h)

Function

Controls various settings for timer *n*.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-24 TRGSEL	Trigger Select Selects the trigger. The valid values for TRGSEL depend on the configuration of Parameter (PARAM): • If TRGSRC = 1, the valid values for *n* depend on the settings of PARAM[PIN], PARAM[TIMER], and PARAM[SHIFTER]. • If TRGSRC = 0, the valid values for *n* depend on PARAM[TRIGGER]. See the chip-specific FLEXIO information for external trigger selection. **NOTE** For a pin, *n* = 0 to 31, for a shifter, *n* = 0 to 7, and for a timer, *n* = 0 to 7. If TRGSRC = 0, configure the trigger selection as *n* = external trigger *n* input. If TRGSRC = 1, you can configure the internal trigger to select an input pin as 2×*n* = pin *n* input. If TRGSRC = 1, you can configure the internal trigger to select a shifter or timer signal as: • 4×*n* + 1 = shifter *n* status flag • 4×*n* + 3 = timer *n* trigger output Following are the values for expanded internal trigger selection (TRGSRC = 1):

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 0000 = Pin 0 • 0001 = Shifter 0 flag • 0010 = Pin 1 • 0011 = Timer 0 trigger • 0100 = Pin 2 • 0101 = Shifter 1 flag • 0110 = Pin 3 • 0111 = Timer 1 trigger • ... • This continues up to pin 31, shifter 7, and timer 7.
23 TRGPOL	Trigger Polarity Specifies whether the trigger is active high or active low. 0b - Active high 1b - Active low
22 TRGSRC	Trigger Source Specifies whether the selected trigger source is external or internal. 0b - External 1b - Internal
21-18 —	Reserved
17-16 PINCFG	Timer Pin Configuration Configures the direction of the timer pin. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register. **NOTE** When you initially configure PINCFG as 11b, FLEXIO may briefly drive the pin low. To avoid this, configure PINCFG as 10b along with the rest of the Control register and then perform a subsequent write to set the value of PINCFG as 11b. Likewise, when changing the value of PINCFG from 11b to 00b, you must perform an initial write to set PINCFG as 10b, and then perform a subsequent write to update the rest of the Control register with PINCFG as 00b. 00b - Timer pin output disabled 01b - Timer pin open-drain or bidirectional output enable 10b - Timer pin bidirectional output data 11b - Timer pin output

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-13 —	Reserved
12-8 PINSEL	Timer Pin Select Selects the pin that is used by the timer input or output. PINSEL = i selects the FXIO_Di pin. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register.
7 PINPOL	Timer Pin Polarity Specifies the timer pin polarity. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register. 0b - Active high 1b - Active low
6 PININS	Timer Pin Input Select Specifies what selects the timer pin input. If this field is 1, the timer input pin is different from the timer output pin. PINSEL must select an even-numbered pin when this field is 1, which means that the output pin is even-numbered and input pin is odd-numbered. 0b - PINSEL selects timer pin input and output 1b - PINSEL + 1 selects the timer pin input; timer pin output remains selected by PINSEL
5 ONETIM	Timer One Time Operation Configures the timer to perform a single enable or disable iteration. Clear the timer status flag for the timer to be enabled again. 0b - Generate the timer enable event as normal 1b - Block the timer enable event unless the timer status flag is clear
4-3 —	Reserved
2-0 TIMOD	Timer Mode Specifies the timer mode: • In 8-bit baud counter mode, the lower 8 bits of the counter and compare register are used to configure the baud rate of the timer shift clock. The upper 8 bits are used to configure the shifter bit count. • In 8-bit PWM high mode, the lower 8 bits of the counter and compare register are used to configure the high period of the timer shift clock. The upper 8 bits are used to configure the low period of the timer shift clock. The shifter bit count is configured using another timer or external signal. • In 16-bit counter mode, the full 16 bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count. • In 16-bit counter disable mode, the full 16 bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• In 8-bit word counter mode, the lower 8 bits of the counter and compare register are used to configure the shifter bit count. The upper 8 bits are used to configure the shifter word count. • In 8-bit PWM low mode, the lower 8 bits of the counter and compare register are used to configure the low period of the timer shift clock. The upper 8 bits are used to configure the high period of the timer shift clock. Use another timer or external signal to configure the shifter bit count. • In 16-bit input capture mode, the inverted value of the 16-bit counter is latched into the compare register when a timer counter disable condition is detected (as configured by TIMCFG*n*[TIMDIS]). This also sets the timer status flag. The timer counter is immediately restarted from FFFFh. 000b - Timer disabled 001b - Dual 8-bit counters baud mode 010b - Dual 8-bit counters PWM high mode 011b - Single 16-bit counter mode 100b - Single 16-bit counter disable mode 101b - Dual 8-bit counters word mode 110b - Dual 8-bit counters PWM low mode 111b - Single 16-bit input capture mode

71.7.1.34 Timer Configuration (TIMCFG0 - TIMCFG7)

Offset

For n = 0 to 7:

Register	Offset
TIMCFGn	480h + (n × 4h)

Function

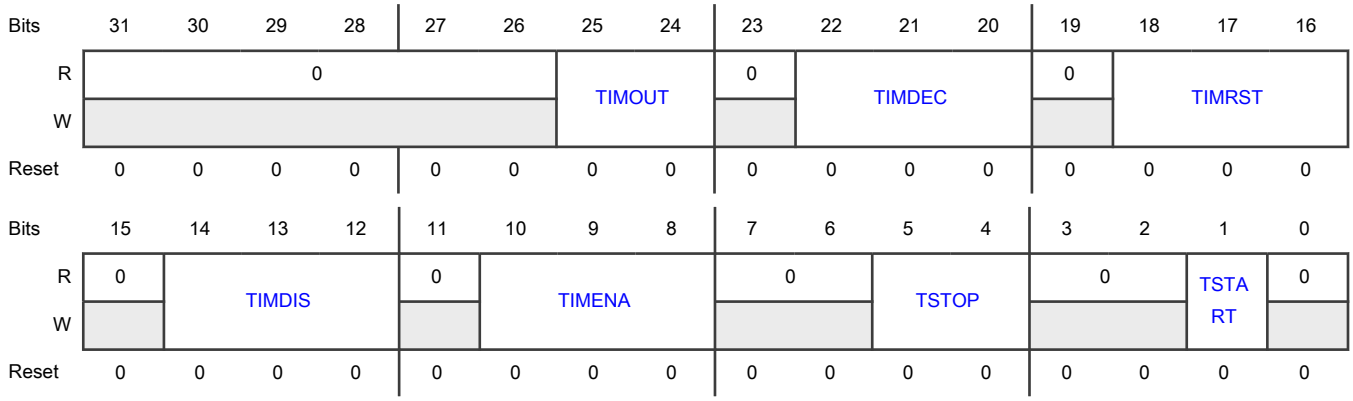
Controls various aspects of timer configuration.

The options to enable or disable the timer using the timer *n* - 1 enable or disable are reserved when *n* is evenly divisible by 4 (timer 0, for example).

NOTE

The pin and trigger level and edges specified in this register refer to the signal state after being modified by the settings of TIMCTL*n*[PINPOL] and TIMCTL*n*[TRGPOL]. For example, "trigger low" means that a trigger is actually at logic level 1 if TIMCTL*n*[TRGPOL] is 1 (active low).

Diagram



Fields

Field	Function
31-26 —	Reserved
25-24 TIMOUT	Timer Output Configures the initial state of the timer output and whether it is affected by the timer reset. 00b - Logic one when enabled; not affected by timer reset 01b - Logic zero when enabled; not affected by timer reset 10b - Logic one when enabled and on timer reset 11b - Logic zero when enabled and on timer reset
23 —	Reserved
22-20 TIMDEC	Timer Decrement Configures the source of the timer decrement and that of the shift clock. 000b - Decrement counter on FLEXIO clock; shift clock equals timer output 001b - Decrement counter on trigger input (both edges); shift clock equals timer output 010b - Decrement counter on pin input (both edges); shift clock equals pin input 011b - Decrement counter on trigger input (both edges); shift clock equals trigger input 100b - Decrement counter on FLEXIO clock divided by 16; shift clock equals timer output 101b - Decrement counter on FLEXIO clock divided by 256; shift clock equals timer output 110b - Decrement counter on pin input (rising edge); shift clock equals pin input 111b - Decrement counter on trigger input (rising edge); shift clock equals trigger input
19 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-16 TIMRST	Timer Reset Configures the condition that causes the timer counter (and optionally the timer output) to be reset. In 8-bit counter mode, the timer reset only resets the lower 8 bits that configure the baud rate. In all other modes, the timer reset resets full 16 bits of the counter. 000b - Never reset timer 001b - Timer reset on timer output high. 010b - Timer reset on timer pin equal to timer output 011b - Timer reset on timer trigger equal to timer output 100b - Timer reset on timer pin rising edge 101b - Reserved 110b - Timer reset on trigger rising edge 111b - Timer reset on trigger rising or falling edge
15 —	Reserved
14-12 TIMDIS	Timer Disable Configures the condition that causes the timer to be disabled and stop decrementing. 000b - Timer never disabled 001b - Timer disabled on timer n-1 disable 010b - Timer disabled on timer compare (upper 8 bits match and decrement) 011b - Timer disabled on timer compare (upper 8 bits match and decrement) and trigger low 100b - Timer disabled on pin rising or falling edge 101b - Timer disabled on pin rising or falling edge provided trigger is high 110b - Timer disabled on trigger falling edge 111b - Reserved
11 —	Reserved
10-8 TIMENA	Timer Enable Configures the condition that causes the timer to be enabled and start decrementing. 000b - Timer always enabled 001b - Timer enabled on timer n-1 enable 010b - Timer enabled on trigger high 011b - Timer enabled on trigger high and pin high

Table continues on the next page...

Table continued from the previous page...

Field	Function
	100b - Timer enabled on pin rising edge 101b - Timer enabled on pin rising edge and trigger high 110b - Timer enabled on trigger rising edge 111b - Timer enabled on trigger rising or falling edge
7-6 —	Reserved
5-4 TSTOP	Timer Stop Specifies whether the stop bit is enabled. The stop bit can be added on a timer compare (between each word) or on a timer disable. When stop bit is enabled, configured shifters output the contents of the stop bit when the timer is disabled. When stop bit is enabled on timer disable, the timer remains disabled until the next rising edge of the shift clock. If configured for both timer compare and timer disable, only one stop bit is inserted on timer disable. 00b - Disabled 01b - Enabled on timer compare 10b - Enabled on timer disable 11b - Enabled on timer compare and timer disable
3-2 —	Reserved
1 TSTART	Timer Start Specifies whether the start bit is enabled. If it is enabled, configured shifters output the contents of the start bit when the timer is enabled. The timer counter reloads from the compare register on the first rising edge of the shift clock. 0b - Disabled 1b - Enabled
0 —	Reserved

71.7.1.35 Timer Compare (TIMCMP0 - TIMCMP7)

Offset

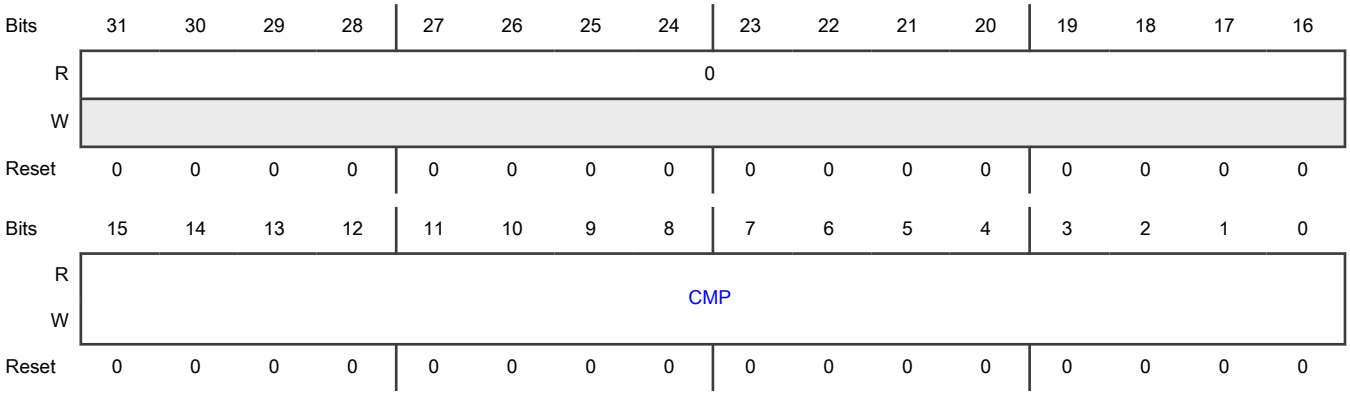
For n = 0 to 7:

Register	Offset
TIMCMPn	500h + (n × 4h)

Function

Contains the timer compare value.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CMP	Timer Compare Value Loads into the timer counter when the timer is first enabled, when the timer is reset, and when the timer decrements down to zero. In 8-bit baud counter mode, the lower 8 bits configure the baud rate divider as (CMP[7:0] + 1) × 2. The upper 8 bits configure the number of bits in each word as (CMP[15:8] + 1) ÷ 2. In 8-bit PWM high mode, the lower 8 bits configure the high period of the output to (CMP[7:0] + 1) and the upper 8 bits configure the low period of the output to (CMP[15:8] + 1). In 16-bit counter mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) as (CMP[15:0] + 1) × 2. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word as (CMP[15:0] + 1) ÷ 2. In 16-bit counter disable mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) as (CMP[15:0] + 1) × 2. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word as (CMP[15:0] + 1) ÷ 2. In 8-bit word counter mode, the lower 8 bits configure the number of bits in each word as (CMP[7:0] + 1) ÷ 2. The upper 8 bits configure the number of words to transfer equal to (CMP[15:8] + 1) ÷ 2. In 8-bit PWM low mode, the lower 8 bits configure the low period of the output to (CMP[7:0] + 1) and the upper 8 bits configure the high period of the output to (CMP[15:8] + 1). In 16-bit input capture mode, the compare register is updated with the inverse of the timer counter value whenever the timer status flag is set. You must read this register only when the timer status flag is set.

71.7.1.36 Shifter Buffer Nibble Byte Swapped (SHIFTBUFNBS0 - SHIFTBUFNBS7)

Offset

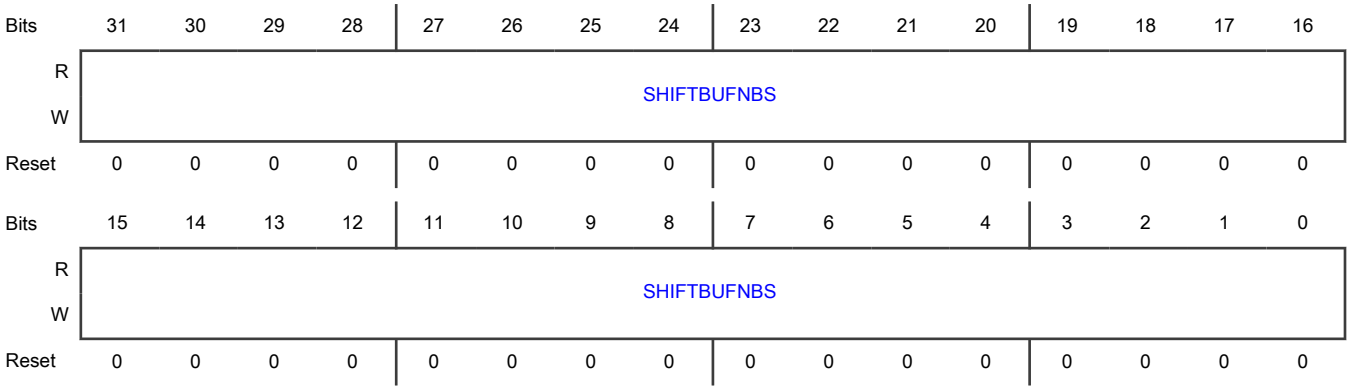
For n = 0 to 7:

Register	Offset
SHIFTBUFNBSn	680h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is nibble-swapped within each byte.

Diagram



Fields

Field	Function
31-0 SHIFTBUFNBS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are nibble-swapped within each byte. Reads return {SHIFTBUF[27:24], SHIFTBUF[31:28], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[3:0], SHIFTBUF[7:4]}.

71.7.1.37 Shifter Buffer Halfword Swapped (SHIFTBUFHWS0 - SHIFTBUFHWS7)

Offset

For n = 0 to 7:

Register	Offset
SHIFTBUFHWSn	700h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is halfword-swapped.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFHWS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFHWS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SHIFTBUFHWS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are halfword-swapped. Reads return {SHIFTBUF[15:0], SHIFTBUF[31:16]}.

71.7.1.38 Shifter Buffer Nibble Swapped (SHIFTBUFNIS0 - SHIFTBUFNIS7)

Offset

For n = 0 to 7:

Register	Offset
SHIFTBUFNISn	780h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is nibble-swapped.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFNIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFNIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SHIFTBUFNIS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are nibble-swapped. Reads return {SHIFTBUF[3:0], SHIFTBUF[7:4], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[27:24], SHIFTBUF[31:28]}.

71.7.1.39 Shifter Buffer Odd Even Swapped (SHIFTBUFOES0 - SHIFTBUFOES7)

Offset

For n = 0 to 7:

Register	Offset
SHIFTBUFOESn	800h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it has odd and even bits partitioned separately.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFOES															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFOES															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SHIFTBUFOES	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register have the odd and even bits partitioned separately. Only 32-bit accesses are supported for this register. Reads return {SHIFTBUF[31], SHIFTBUF[29], SHIFTBUF[27],SHIFTBUF[25],SHIFTBUF[23],SHIFTBUF[21],SHIFTBUF[19],SHIFTBUF[17],SHIFTBUF[15],SHIFTBUF[13],SHIFTBUF[11],SHIFTBUF[9], SHIFTBUF[7], SHIFTBUF[5], SHIFTBUF[3], SHIFTBUF[1], SHIFTBUF[30], SHIFTBUF[28], SHIFTBUF[26], SHIFTBUF[24], SHIFTBUF[22], SHIFTBUF[20], SHIFTBUF[18], SHIFTBUF[16], SHIFTBUF[14], SHIFTBUF[12], SHIFTBUF[10], SHIFTBUF[8], SHIFTBUF[6], SHIFTBUF[4], SHIFTBUF[2], SHIFTBUF[0]}.

71.7.1.40 Shifter Buffer Even Odd Swapped (SHIFTBUFEOS0 - SHIFTBUFEOS7)

Offset

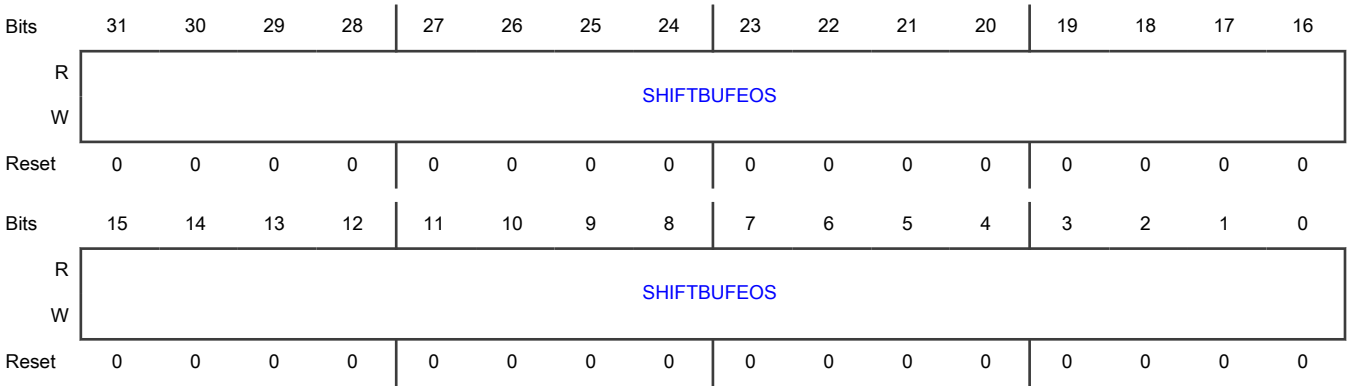
For n = 0 to 7:

Register	Offset
SHIFTBUFEOSn	880h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, with even and odd bits partitioned separately.

Diagram



Fields

Field	Function
31-0 SHIFTBUFEOS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register have the even and odd bits partitioned separately. Only 32-bit accesses are supported for this register. Reads return {SHIFTBUF[30], SHIFTBUF[28], SHIFTBUF[26], SHIFTBUF[24], SHIFTBUF[22], SHIFTBUF[20], SHIFTBUF[18], SHIFTBUF[16], SHIFTBUF[14], SHIFTBUF[12], SHIFTBUF[10], SHIFTBUF[8], SHIFTBUF[6], SHIFTBUF[4], SHIFTBUF[2], SHIFTBUF[0], SHIFTBUF[31], SHIFTBUF[29], SHIFTBUF[27],SHIFTBUF[25],SHIFTBUF[23],SHIFTBUF[21],SHIFTBUF[19],SHIFTBUF[17],SHIFTBUF[15],SHIFTBUF[13],SHIFTBUF[11],SHIFTBUF[9], SHIFTBUF[7], SHIFTBUF[5], SHIFTBUF[3], SHIFTBUF[1]}.

71.7.1.41 Shifter Buffer Halfword Byte Swapped (SHIFTBUFHBS0 - SHIFTBUFHBS7)

Offset

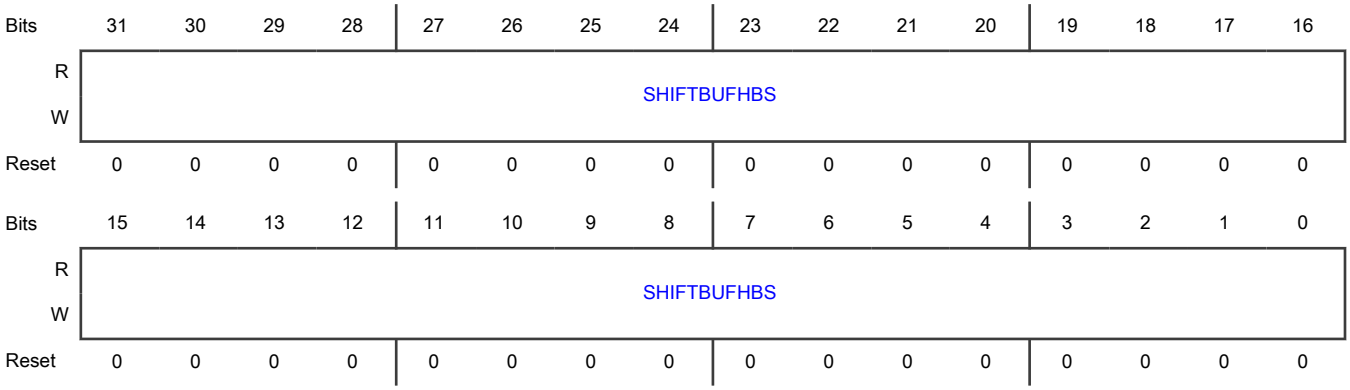
For n = 0 to 7:

Register	Offset
SHIFTBUFHBSn	900h + (n × 4h)

Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is halfword byte-swapped.

Diagram



Fields

Field	Function
31-0 SHIFTBUFHBS	Shift Buffer Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are halfword byte-swapped. Reads return {SHIFTBUF[23:16], SHIFTBUF[31:24], SHIFTBUF[7:0], SHIFTBUF[15:8]}.

Chapter 72
Improved Inter-Integrated Circuit (I3C)

72.1 Chip-specific I3C information

Table 695. Reference links to related information

Topic	Related module	Reference
Full description	I3C	I3C
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

72.1.1 Module instances

This device has two instances of the I3C module, I3C0 and I3C1.

72.1.2 High-Keeper methods

This device supports all High-Keeper methods.

72.2 Overview

I3C is a communications processor that improves upon the use and power of I2C, and provides an alternative to SPI for mid-speed applications.

The I3C bus protocol supports:

- In-band interrupts (IBI): these interrupts move from target to controller without extra wires, and the controller knows which target sent the interrupt
- Common command codes (CCC)
- Dynamic addressing
- Multi-controller and multi-drop
- Hot-Join (HJ)
- I2C compatibility

I3C supports all required and many optional features of the MIPI Alliance Specification for I3C, v1.0, except for ternary data rates (HDR-TSP and HDR-TSL). See Features for more information.

72.2.1 Block diagram

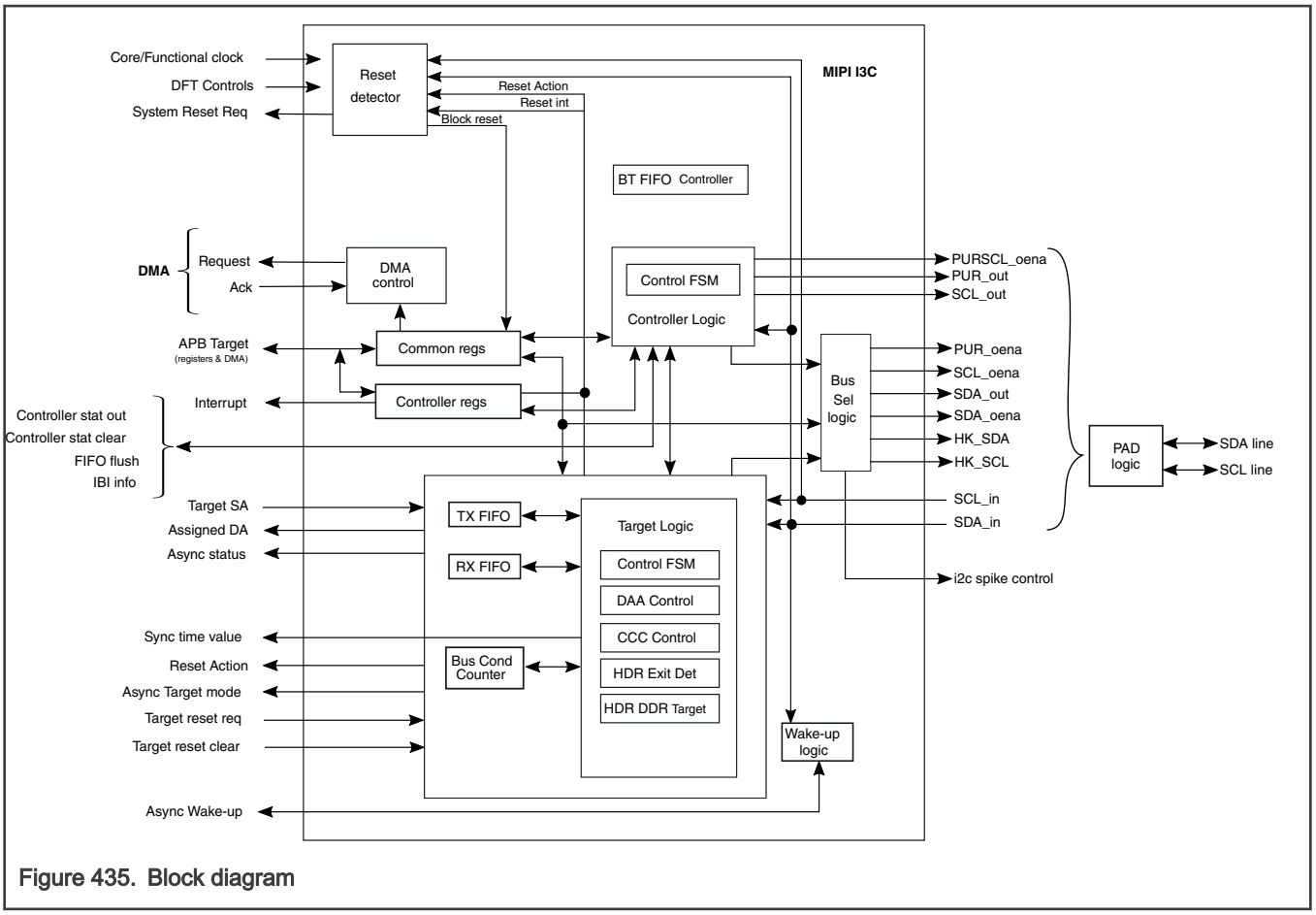


Figure 435. Block diagram

72.2.2 Features

- Two-wire multidrop bus that is capable of 12.5 MHz clock speeds with up to 11 devices:
  - Uses standard pads with 4 mA drive.
  - Dynamically assigns target addresses, and targets do not require static addresses (SAs). However, targets can have an I2C static address assigned at startup, so the target can operate on an I2C bus. By default, I3C supports 7-bit I2C-style addresses.
  - Allows targets to use the inbound SCL clock as the peripheral clock (instead of the clock from the controller) so that devices can have slow or inaccurate clocks internally.
  - Allows simple targets, such as temperature sensors, to have no internal clock.
  - I3C controller supports handoff from Open-Drain to Push-Pull mode for ACK to data transfer.
  - Generally, the controller terminates the read, but for I3C, the target can also end the read.
- IBIs that allow targets to send notifications to a controller:
  - Can be equivalent to a separate GPIO, but can also be directly data-bearing.
  - Can be prioritized. When multiple targets send interrupts to a controller at the same time, the order is resolved. Dynamic addresses (DAs) establish the priority of the targets, so the controller controls the priority of the targets.

    Targets with lower-value DAs are higher-priority level IBIs.

  - Can start interrupts even when the controller is not active on the bus. A free-running clock is not required, but starting an interrupt requires a Bus Available condition.
  - Can resolve an initial event via a time-stamping option, not requiring an interrupt.
- Built-in commands for applications created in software or firmware by using the register interface maintained in a separate space. These commands do not collide with normal controller-to-target messages and:
  - Control bus behavior, modes and states, low-power state, inquiries, and more.
  - Have an extra room for new built-in commands for other groups.
- Organized forms of multicontroller modes that are secondary controllers using clean handoffs between different controllers.
- Hot-Join onto I3C bus allows devices to connect to the bus later than when the bus starts. This:
  - Enables a device or module to access the I3C bus after power up or when physically inserted onto the I3C bus.
  - Provides a clean method for notification when new devices or modules access the I3C bus.
- I3C can use both I2C and I3C buses:
  - I3C supports specific legacy I2C devices on the bus.
  - I3C target devices can operate on I2C buses.
  - I3C supports bridging to I2C, SPI, UART, and other buses.
  - Special mode for old-style I2C buses (no targets) with clock stretching.
- Higher data rate modes are available:
  - I3C has a high data rate: double data rate (HDR-DDR) mode, which is double the data rate of SDR .
  - Only the controller and the specific target must support the higher data rate (the other targets can ignore it).

I3C supports most of the I3C features (see Target Capabilities (SCAPABILITIES) and Target Capabilities 2 (SCAPABILITIES2)), except for the ternary data rates (HDR-TSP and HDR-TSL) and peer-to-peer messaging.

72.3 Functional description

72.3.1 Operating modes

This section describes all functional operation modes of the I3C module.

72.3.1.1 Target and controller roles for I3C

The I3C protocol defines these roles for devices on the I3C bus:

- Main controller: initially configures the I3C bus and serves as the first active controller
- Secondary controller: accepts the controller role from any active controller to become the new active controller (the secondary controller may then pass the controller role along: either back to the previous active controller, or on to any other controller-capable device)
- Target: responds to commands from any I3C controller
- I2C target: responds to commands from any controller

The I3C peripheral contains both controller and target components and can be configured to be either controller or target, or as target with secondary controller capability. However, if I3C is chosen to be a controller, then the I3C peripheral supports Target mode to facilitate handoffs to multiple controllers.

In general, any I3C controller can be a controller or a target because a controller becomes a target when giving over control to another controller. The only exceptions to this rule occur when using point-to-point communication or when using a controller that never shares controllership.

72.3.1.1.1 Controller requirements

Controller mode uses software that supports the requirements of the controller (including as a secondary controller):

- Managing the enter dynamic address assignment (ENTDAA) assigning dynamic addresses (DAs) to each target. The I3C peripheral supports this process but requires you to make choices.
- Driving the serial data (SDA) signal in both Open-Drain and Push-Pull modes. After START command, SDA is in Open-Drain mode and after Repeated START command, it is in Push-Pull mode:
  - SDA is subject to arbitration because both controller and target can drive the SDA line (arbitration is also useful for handoff from controller to target).
  - The ninth bit of SDA controller write data is an odd parity bit.
- Building a table of targets and their capabilities to control which actions and commands may be sent to the targets.
- Managing requests such as IBI and controller requests (handoff).
- Adjusting clock speed or write-to-read timing to match the limitations of the target. This adjustment can be done in hardware using dividers and uneven duty cycles, but you must decide how.
- Adjusting the maximum data length.
- Being a target after the controller role handoff to another controller-capable device.

The controller needs an accurate clock capable of running at a frequency that is the multiple of a frequency between 11 MHz and 12.5 MHz. Following are the possible clock frequencies:

- 24 MHz (multiple of 12 MHz)
- 48 MHz (multiple of 12 MHz)
- 133 MHz
- 160 MHz

If required to use a lower I3C SCL value, for example as small as 5 MHz (although the lower value does not support a mixed bus system):

- The controller clock can be a multiplier of that value.
- A lower SCL can also be achieved using a higher PPBAUD value.

72.3.1.1.2 I3C target acts like I2C target on I3C buses

If the target is assigned an I2C static address, the I3C target acts like an I2C device when it first powers on. If the I3C target is placed on an I2C bus with an I2C controller, then the I3C target stays in I2C mode and operates normally. The software is aware that the I3C target is in I2C mode because:

- There has not been an interrupt (see SSTATUS[DACHG]) indicating that a dynamic address was assigned.

For full I2C support of Fast mode (Fm) and Fast-mode Plus (Fm+), the pads must support a 50-ns spike filter. This filter must be turned off when the I3C 7Eh broadcast address is received (indicating an I3C controller). You can turn off the spike filters via hardware using the raw net indicating that the address was received, or via software. I3C does not need a 50-ns spike filter to operate on an I2C bus. Therefore, the spike filter is not a requirement for the I3C peripheral.

Depending on the configuration, the module supports most I2C features. Supported features include an extended 10-bit address, DeviceID, and software reset. See Target Capabilities (SCAPABILITIES) and Target Capabilities 2 (SCAPABILITIES2) for more information.

72.3.1.1.3 How a target rejoins the I3C bus

When a target tries to rejoin the I3C bus following a power-up or hard reset, the target needs a new dynamic address (DA). The target can rejoin the I3C bus in these ways:

- If the DA is lost, Hot-Join is used.

- If the DA is retained in the peripheral (for example, with state retention flip-flops), SCONFIG[OFFLINE] is used.

When SCONFIG[SLVENA] is 1, SCONFIG[OFFLINE] may become 1. This setting causes the peripheral to rejoin the bus safely. It does so by ensuring that the I3C bus is not in HDR mode, using the same approach as TE0 or TE1 exit. The peripheral waits for the HDR exit pattern from the controller, or for 60 µs of SCL and SDA lines not changing, whichever occurs first.

After using SCONFIG[OFFLINE], the I3C peripheral cannot safely use IBI until SSTATUS[STOP] becomes 1. This status ensures that the next START is safe to use for IBI.

If the application has to perform an IBI, it must wait for SSTATUS[STOP] to become 1, or for SCL and SDA lines to remain high for 200 µs. You can perform this process using the SSTATUS and SINTSET controls:

- If Target Status (SSTATUS) indicates that the bus is not busy and the peripheral interrupts on START or STOP, use a timer to measure 200 µs. If the timer finishes with no START or STOP, it is safe to use IBI.
- If a START causes an interrupt, then the timer must be turned off and the application must wait for a STOP.
- If a STOP causes an interrupt, it is safe to use IBI.

72.3.1.2 Using the I3C controller for I2C and I3C

The I3C controller operates with the following set of built-in capabilities with the application flow:

- Built-in enter dynamic address assignment (ENTDAA) mechanism to simplify the assignment of dynamic addresses to targets. This feature is used when set dynamic address from static address (SETDASA) is not available.

  NOTE

  When SETDASA CCC is available, use SETDASA CCC before using ENTDAA CCC; all targets without assigned dynamic addresses respond to ENTDAA CCC.

- Request for START + address with IBI support, including both I2C and I3C modes.
- SDR write flow via FIFO, with automatic parity in I3C and ACK or NACK detection in I2C.
- SDR read flow via the FIFO, with an automatic NACK generator for I2C, and optional use of a terminate generator for I3C.
- Request for Repeated START + address or STOP when finished with the previous request, including both I2C and I3C.
- Auto-IBI mode, which responds immediately to a target-initiated IBI and can be used when in Sleep or Deep Sleep mode.
- Special message mode for SDR and DDR to simplify DMA use.
- Automated SCL, SDA, pullup, and High-Keeper controls.
- I2C and I3C frequency and duty cycle configurations from functional clock.

  NOTE

  You must configure Controller Configuration (MCONFIG) and Controller In-band Interrupt Registry and Rules (MIBIRULES) before using the controller.

72.3.1.3 Protocol modes and states

The following modes are activated in I3C:

- I2C mode is the default mode on startup:
  - If no I2C static address is used, this mode has no effect other than to track frames looking for I3C transitional frames.
  - If an I2C static address is used, the device can interact with the I2C bus controller using the address.
- When in I2C mode, I3C Transitory Frame mode occurs whenever the I3C broadcast address is received (7Eh). In particular:
  - This mode occurs when 7Eh is broadcast followed by SETDASA from the controller. If there is a static address match or 01h, the target is assigned a dynamic address and it enters I3C SDR mode.

— This mode occurs when 7Eh is broadcast followed by ENTDAA from the controller. If the target can send a 48-bit provisioned ID (48b ID), a dynamic address is assigned, and the target enters I3C SDR mode.

— If a Hot-Join arbitrated event occurs (sending 02h when the controller generates another address such as 7Eh), there is a conceptual Hot-Join mode. ENTDAA or SETDASA sets a dynamic address to resolve the event, and the target enters I3C SDR mode.

NOTE

An I3C target can operate as a normal I2C target only when it has a static address. Otherwise, it matches nothing in I2C and waits for the aforementioned events.

- I3C SDR mode is the standard mode after I3C activates, which is defined by a dynamic address being assigned. This mode is the resting mode of I3C, and all devices are normally in SDR mode:
  — RSTDAA CCC causes an exit from SDR mode and a return to I2C mode. This transition is not normally used.

    RSTDAA can be issued to ensure all I3C targets participate in an upcoming dynamic address assignment process, before the I3C controller starts assigning the target address.

  — SETNEWDA does not affect the SDR mode; it only changes the dynamic address of an I3C device that already had a dynamic address assigned.

  — When in SDR mode, the device ignores messages to or from its original I2C static address.
- The I3C CCC command submode of type Direct or Broadcast. Following are the options to exit the CCC:
  — Exit the Broadcast CCC submode by a Repeated START or by a STOP.

  — Exit the Direct CCC submode by a 7Eh broadcast address after a Repeated START or by a STOP.

  — I3C dynamic address assignment (DAA) CCC command enters DAA mode. This mode is a special mode for dynamic addressing. Use STOP to exit. If a target has a dynamic address, the command is ignored.
- I3C SETDASA CCC command enters Static Address Match submode, which allows matching the static address of the device (if any). The command is ignored when a target has a dynamic address, or when the target in I2C mode has no static address.

NOTE

The special point-to-point address is also matched.

- I3C HDR modes are activated by ENTHDR*n* CCC commands, where *n* represents the type of HDR (0 to 7). This mode is valid until an HDR exit pattern is detected, irrespective of whether HDR mode for a target is supported.

  Exit-pattern detection must always be on to prevent errors. Alternatively, you could set it to be activated only on ENTHDR.
- Internal states such as IBI or no-IBI and Low-Power mode are flagged internally (see SSTATUS[EVDET] for more information). These states are exported to the application and affect the engine to restrain it from performing a prohibited operation.

72.3.1.4 Address match

For an address match to occur, the target matches the I3C broadcast address and either the I2C-style static address or I3C dynamic address (see Table 696).

The address match is inactive (just listening) for all Repeated START commands and also for the START command, unless an IBI, CR, or Hot-Join has been activated. If inactive, it simply tries to match. If not matched, it waits for a Repeated START or STOP command. If an IBI, CR, or Hot-Join has been activated, the arbitration mechanism is used for START (but never for a Repeated START).

When matching the 7Eh broadcast address, it looks for a CCC until the next Repeated START or STOP command.

Table 696. Address match conditions

For	Match condition occurs when	Notes
I3C broadcast address	The address is 111 1110b (written as 7Eh)	
I2C-style static address	• The device has a static address. • The device is not in I3C mode.	When in I3C mode, the address matches only until the ENTDAA, SETDASA, or SETAASA command has a dynamic address assigned. **NOTE** SETDASA matches the static address or the special one-controller-to-one-target point-to-point address.
I3C dynamic address	It is assigned by ENTDAA or SETDASA or modified by SETNEWDA	

72.3.2 Operations

This section describes the operations of the module.

72.3.2.1 Reading and writing I2C messages using the normal method

1. Set up interrupts using the following fields of Controller Interrupt Set (MINTSET):
   - MCTRLDONE: Indicates when the module completes an MCTRL request.
   - COMPLETE: Indicates when data is finished sending or being received.
   - RXPEND: Used for read operations. Works with Controller Data Control (MDATACTRL) to set the FIFO trigger. You can also use this field to allow DMA to read out data.
   - TXNOTFULL: Used for write operations. Works with Controller Data Control (MDATACTRL) to set the FIFO trigger. You can also use this field to allow DMA to supply data.
   - IBIWON: Indicates that an IBI, CR, or HJ has won the arbitration on a header address.
   - Additionally, Controller Errors and Warnings (MERRWARN) indicates the causes of errors and warnings.
2. Configure the following fields in Controller Control (MCTRL) simultaneously:
   a. Write 1 to REQUEST (EmitStartAddr).
   b. Write 1 to TYPE (I2C).
   c. Configure IBIRESP to respond to IBIs in your chosen manner.
   d. Write 1 to DIR for read, or write 0 for write.
   e. Write the static address of the I2C target to ADDR.
   f. Configure RDTERM to the maximum length for read operations, to auto-terminate, or set it to STOP as the data is read out. For example, write 1 (with MCTRL[REQUEST] = 0) to stop after the next character.
3. Write or read the data.

- For write operations, write to Controller Write Data Byte (MWDATAB) for each byte before the last byte, and then write to Controller Write Data Byte End (MWDATABE) for the last byte:
  - — You can perform or start this operation before setting up interrupts.
  - — If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level. This interrupt allows the application to provide more data or to use DMA.
- For read operations, wait for RXPEND, and then read out data via Controller Read Data Byte (MRDATAB). DMA may also be used.

4. After message transfer is complete (MSTATUS[COMPLETE] = 1), the message may end with a STOP, or a new message may start with a Repeated START:
   a. Write 2 to MCTRL[REQUEST] (EmitStop to STOP). Then wait for the MCTRLDONE status to be asserted for its completion. (When sending STOP in I2C mode, MCONFIG[ODSTOP] and MCTRL[TYPE] must be 1.)
   b. Write 1 to MCTRL[REQUEST] (EmitStartAddr to restart). IBI is not possible in this case.

NOTE

I2C Fm and I2C Fm+ modes are supported for legacy I2C devices as per the I3C standard specifications. See I2C configuration to meet timing requirements for Fm and Fm+ modes for more information.

72.3.2.2 Reading and writing I3C messages using the normal methods (SDR and HDR-DDR)

The normal method for I3C is the same as the method for I2C with a few differences:

1. Set up interrupts using the following fields of Controller Interrupt Set (MINTSET):
   - MCTRLDONE: Indicates when the I3C module completes an MCTRL request.
   - COMPLETE: Indicates when data is finished sending or being received.
   - RXPEND: Used for read operations. Works with Controller Data Control (MDATACTRL) to set the FIFO trigger. You can also use it to allow DMA to read out bytes.
   - TXNOTFULL: Used for write operations. Works with Controller Data Control (MDATACTRL) to set the FIFO trigger. You can also use it to allow DMA to supply bytes.
   - IBIWON: Indicates that an IBI, CR, or HJ has won the arbitration on a header address.
   - Additionally, Controller Errors and Warnings (MERRWARN) indicates the causes of errors and warnings.
2. Set up Controller Control (MCTRL).
   - Option 1: Configure the following fields of Controller Control (MCTRL) simultaneously in this way:
     a. Write 1 to REQUEST (EmitStartAddr).
     b. Write 0 to TYPE for I3C SDR mode or write 2 for DDR mode.
     c. Configure IBIRESP to respond to IBIs in your chosen manner.
     d. Write 1 to DIR for read, or write 0 for write.
     e. Write the dynamic address of the I3C target to ADDR.
     f. For read operations, you can configure RDTERM to the maximum length to terminate automatically.
     g. For write operations, prewriting the data (MWDATAB or MWDATAH) is preferred to ensure that there are no time delays waiting for the data.

        For DMA with MCTRL, use Controller Write Byte Data 1 (to Bus) (MWDATAB1).

NOTE

HDR-DDR mode requires writing an 8-bit command value for read or write. This value must be written into the TX FIFO via Controller Write Data Byte (MWDATAB). The END field is not used for this byte.

- Option 2: This option is preferred when stopped (bus free condition) in SDR mode, and not in HDR-DDR mode. It allows any target to issue an IBI and avoids collisions with an IBI address. Also, it is faster (when the MSB of the dynamic address is always 0).

  Configure the following fields of Controller Control (MCTRL) in this way:

  a. Write 1 to REQUEST (EmitStartAddr). No prewritten transmit data can be in the FIFO.
  b. Write 0 to TYPE (I3C).
  c. Configure IBIRESP to respond to IBIs in your chosen manner.
  d. Write 0 to DIR.
  e. Write 7Eh to ADDR.
  f. Wait for MCTRLDONE (via interrupt, for example), then proceed as in option 1. The option 2 method has advantages for IBIs when 7Eh is sent on START (but not on Repeated START conditions).

3. Write or read the data.
   - For write operations, write to MWDATAB for each byte before the last byte, and then write to Controller Write Data Byte End (MWDATABE) for the last byte. For HDR-DDR, the byte with END must be even (second, fourth, sixth, and so on) because DDR uses byte pairs:
     — You can perform or start this operation before step 1 (REQUEST = 1) of the option 1 method, but not before the option 2 method.
     — If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level. This interrupt allows the application to provide more data or to use DMA.
   - For read operations, wait for RXPEND, and then read out data via Controller Read Data Byte (MRDATAB) or Controller Read Data Halfword (MRDATAH). DMA may also be used. When using DMA, read using the same registers.
4. After message transfer is complete (MSTATUS[COMPLETE] = 1), the message may be ended with STOP (or EXIT in HDR mode). Alternatively, a new message may be started with a Repeated START (or HDR-Restart in HDR mode):
   - In SDR mode, write 2 to MCTRL[REQUEST] (EmitStop to STOP). Then wait for the MCTRLDONE status to be asserted for its completion.
   - For HDR mode, write 6 to MCTRL[REQUEST] (ForceExit) to end HDR mode. Then wait for the MCTRLDONE status to be asserted for its completion. When sending the HDR exit pattern, MCONFIG[ODSTOP] must be 0.
   - Write 1 to MCTRL[REQUEST] (EmitStartAddr to start another message). IBI is not possible in this case.

72.3.2.3 Determining bus types

As shown in the following tables, the meaning of MCTRL[TYPE] changes according to the value of MCTRL[REQUEST].

Table 697. Determining bus types

Value of MCTRL[TYPE]	Meaning when MCTRL[REQUEST] = 1 (EmitStartAddr)	Meaning when MCTRL[REQUEST] = 2 (EmitStop)	Meaning when MCTRL[REQUEST] = 6 (ForceExit)
0	I3C: SDR mode of I3C	I3C: SDR mode of I3C	Exit pattern
1	I2C: standard I2C protocol	I2C: standard I2C protocol	Reserved
2	DDR: HDR-DDR mode of I3C. The bus enters Double Data Rate (DDR) mode (7E and then ENTHDR0), if the module is not in DDR mode already. The first byte	Reserved	Target reset

Table continues on the next page...

Table 697. Determining bus types (continued)

Value of MCTRL[TYPE]	Meaning when MCTRL[REQUEST] = 1 (EmitStartAddr)	Meaning when MCTRL[REQUEST] = 2 (EmitStop)	Meaning when MCTRL[REQUEST] = 6 (ForceExit)
	written to the transmit FIFO must be a command and already in the FIFO. To end DDR mode, use ForceExit.		
3	BT: HDR-BT mode of I3C. If not already in HDR-BT, the bus automatically enters BT mode (7E and then ENTHDR3). You must also configure the MHDRBTCFG register for multilane and CMD rules.	Reserved	Reserved

72.3.2.4 Sending a CCC to I3C targets

The normal common command code (CCC) method uses an I3C write operation with an address of 7Eh (the CCC is the first byte):

- For broadcast type, any remaining bytes are sent with the CCC. This operation ends with a STOP or a Repeated START and 7Eh.
- For direct type, only the CCC byte is sent (or by a CCC and a defining byte, if required by the CCC).

  These bytes are followed by a Repeated START and the address of the I3C target (for SETDASA, this address is its I2C static address). This sequence may be repeated with more Repeated START conditions and addresses until done. It may end in STOP or a Repeated START and 7Eh. After the Repeated START and target address, the values are read or written depending on the CCC.
- I3C provides interrupts (by hardware) for unhandled and handled CCC when received by the target. Its state is reflected in Target Status (SSTATUS).

I3C controllers require to emit a single START, 7E/W sequence with both SCL high and SCL low half periods at full open-drain timing (for example, 200 ns). This sequence allows I3C targets acting as I2C legacy devices to turn off their I2C 50-ns spike filters, if they have them.

NOTE

START, 7E/W is a notation indicating a START command, followed by the 7Eh broadcast address and then by a write command.

After that sequence, addresses following START may be sent with Open-Drain Low (for example, 200 ns) but high of Push-Pull timing (for example, 40 ns). The I3C controller must emit this START, 7E/W sequence with MCONFIG[ODHPP] = 0. ODHPP is Open-Drain High period at Push-Pull speeds, and MCONFIG[ODHPP] is usually 1.

NOTE

- MCONFIG[ODHPP] is ignored when sending a message to an I2C legacy device on an I3C bus.
- Each Repeated START is a new EmitStart request. This request can be chained by an interrupt or pushed by a message model using DMA.

72.3.2.5 IBI handling

An IBI occurs when a target sends its address after a START, and that address is numerically the lowest. That is, it is lower than the address sent by the controller and addresses sent by any other targets. When the controller sends 7Eh, the controller always loses the arbitration.

The IBI can occur unexpectedly when any new START (not a Repeated START) is sent. The IBI can also occur in response to a target pulling SDA low. This condition can occur in one of these ways:

- The controller has set the request to AutoIBI mode, so the IBI occurs automatically. The controller sends 7Eh to allow the target to win the arbitration.
- The application receives the SLVSTART (target START request) interrupt, so it sends 7Eh.

The IBI response is configured by MCTRL[IBIRESP], which can be set to:

- NACK (always reject the IBI).
- ACK (always accept the IBI):
  - You must configure Controller In-band Interrupt Registry and Rules (MIBIRULES) so that the engine knows whether IBI bytes follow.
  - If IBI bytes follow (also known as IBI mandatory byte), then the COMPLETE field does not become 1 when IBIWON becomes 1. RXPEND becomes 1 for one or more bytes in the receive FIFO. When the last byte is received, then the COMPLETE field becomes 1. The controller automatically stops IBI data after 9 bytes (including the mandatory data byte). MCTRL[RDTERM] can be used with MCTRL[REQUEST] = 0 to end the process of receiving data sooner. I3C supports address ACK to mandatory byte transition during IBI from Open_Drain (controller acknowledges the target address) to Push Pull (target sends SDR data).
  - I3C target supports up to 7 bytes (maximum limit is defined by IBIEXT1[MAX]) of extended IBI data following a mandatory data byte. Extended data to send, if any, is present in Extended IBI Data 1 (IBIEXT1) and Extended IBI Data 2 (IBIEXT2).
- Manual (allow the decision to be made by the application on a case-by-case basis):
  - The application rewrites it when stopped, pending an IBI.
  - The application can ACK or NACK the IBI based on the IBI address defined in MSTATUS.
  - This mode selects whether there is an IBI byte when accepting the IBI request from targets.

Accurate timestamping of I3C target data is supported in I3C though Async mode 0. In I3C Asynchronous Timing Control mode, a target device timestamp event occurs within that target. The target notifies the controller about the event by generating an IBI.

72.3.2.6 Assigning dynamic addresses to I3C devices

If dynamic addresses (DAs) are all assigned below 40h (7-bit values from 3Fh down to 03h, except where not allowed), then MIBIRULES[MSB0] can be 1. This setting optimizes the START timing. When dynamic addresses are assigned, you must program Controller In-band Interrupt Registry and Rules (MIBIRULES) based on which I3C target uses IBI bytes (present in SIDEXT[BCR]).

Any SETDASA assignments can be performed via the normal CCC model. These assignments cannot be performed when using the static address for the directed part.

There is a built-in mechanism to process DAA mode:

1. Set up interrupts using the following fields of Controller Interrupt Set (MINTSET):
   - MCTRLDONE: Indicates when a target has sent its ID and BCR or DCR, and a new DA is needed.
   - COMPLETE: Indicates when DAA is done (NACKed by all targets).
   - RXPEND: Indicates the reading of the IDs of the targets. You can also use this field to allow DMA to read out bytes.
   - IBIWON: Indicates when an IBI has occurred (normally there are no IBI conditions when assigning dynamic addresses to the I3C device).
   - Additionally, Controller Errors and Warnings (MERRWARN) indicates the causes of errors and warnings.
2. Configure Controller Control (MCTRL):
   a. Write 4 to MCTRL[REQUEST] (ProcessDAA).
   b. Set MCTRL[IBIRESP] to IBI response, if possible. If not possible, set to the first target assignment.

3. Wait for the MCTRLDONE interrupt when reading ID using the RXPEND interrupt. If MSTATUS[STATE] = 5 (DAA mode) and MSTATUS[BETWEEN] = 1, it is waiting to write a dynamic address:
    a. Write the dynamic address into Controller Write Data Byte (MWDATAB) using bits 6:0, such as 14h for DA = 7'h14.
    b. Write 4 to MSTATUS[STATE] (ProcessDAA again). This option writes the DA, then moves to the next DA.
    c. If MSTATUS[COMPLETE] = 1 and MSTATUS[STATE] = 0, then all targets are assigned.
    d. If MSTATUS indicates NACK after writing a new DA, the DA is not accepted by the target. The next step is to write 2 to MCTRL[REQUEST] (EmitStop) and start over. It is also acceptable to write 4 to MCTRL[REQUEST] (ProcessDAA again).

72.3.2.7 Using Controller Message mode

Controller Message mode is intended for use with DMA although Message mode can also be used by the processor. In Message mode, all writes are to the same location, including control and data. Reads occur from an associated location.

NOTE

The data writes for Message mode work in the same way as the halfword access registers, Controller Write Data Halfword (MWDATAH) and Controller Read Data Halfword (MRDATAH).

To send a message via Single Data Rate (SDR), follow these steps (from DMA or processor):

1. Write the control request to Controller Write Message Control in SDR mode (MWMSG_SDR_CONTROL). This request includes the following items:
    - The address.
    - I2C or I3C.
    - Read or write.
    - The count of bytes to process.
    - How to end (on STOP or ready for Repeated START).
2. Process the data:
    - For write operations, write the rest of the data to Controller Write Message Data in SDR mode (MWMSG_SDR_DATA). Use the DMA trigger (or interrupt) to keep the transmit FIFO full, and the controller stops using the data when it has already consumed the number of bytes configured in the LEN field.
    - For read operations, a DMA trigger (or interrupt) indicates when the RX FIFO is ready to be read out via Controller Read Message in SDR mode (MRMSG_SDR).
3. Select the in-band interrupt (IBI) behavior in MCTRL[IBIRESP].
4. Use END type STOP (MCTRL[REQUEST] = 2) with a zero-length message or by using Controller Control (MCTRL). To exit Message mode with the original message, use END type STOP (MCTRL[REQUEST] = 2) with a zero-length message or by using the MCTRL register.

To send a message via Double Data Rate (DDR), follow these steps (from DMA or processor):

1. Write the control request to Controller Write Message in DDR mode: First Control Word (MWMSG_DDR_CONTROL). This request includes:
    - The count of byte pairs.
    - How to end the message (through HRD exit or to wait for HDR restart).
2. Write the second control data to MWMSG_DDR_CONTROL. This second write includes:
    - The address.
    - Read or write.

- The 7-bit command value.

3. Process the data:
   - For write operations, write the rest of the data to MWMSG_DDR_DATA. Use the DMA trigger (or interrupt) to keep the transmit FIFO full, and the controller stops using the data when the program count MWMSG_DDR_CONTROL register reaches 0.
   - For read operations, a DMA trigger (or interrupt) indicates when the receive FIFO is ready to be read out.
4. Select in-band interrupt (IBI) behavior in MCTRL[IBIRESP].
5. Use END type exit (MCTRL[REQUEST] = 6) with a zero-length message or by using the MCTRL register to exit DDR Message mode with the original message.

72.3.2.8 Handing off controllership to another target and getting it back

To hand off controllership, the controller can perform either of the following actions:

- Wait for a controller request (CR) (that is, MSTATUS[IBIWON] = 1 interrupt), indicating that the CR is using MSTATUS[IBITYPE].
- Push the request manually.

In either case, the controller sends a GETACCMST request, which is a directed GET informing the target that it is being assigned controllership. If the target accepts this request, it returns its dynamic address in bits 7:1 and the negative parity of its dynamic address in bit 0. A STOP must be issued, and MCONFIG[MSTENA] must be set to 2 (switching to Target mode).

To gain controllership, the target sends a CR using Target Control (SCTRL):

- If MCONFIG[MSTENA] is 2, the GETACCCR CCC is ACKed.
- If MCONFIG[MSTENA] is not 2, the GETACCCR CCC is NACKed.

After controllership is granted, MSTATUS[NOWMASTER] becomes 1. The application must enable MINTSET[NOWMASTER], so the application is interrupted when a controllership transfer occurs.

72.3.2.9 Controller engine flow diagram

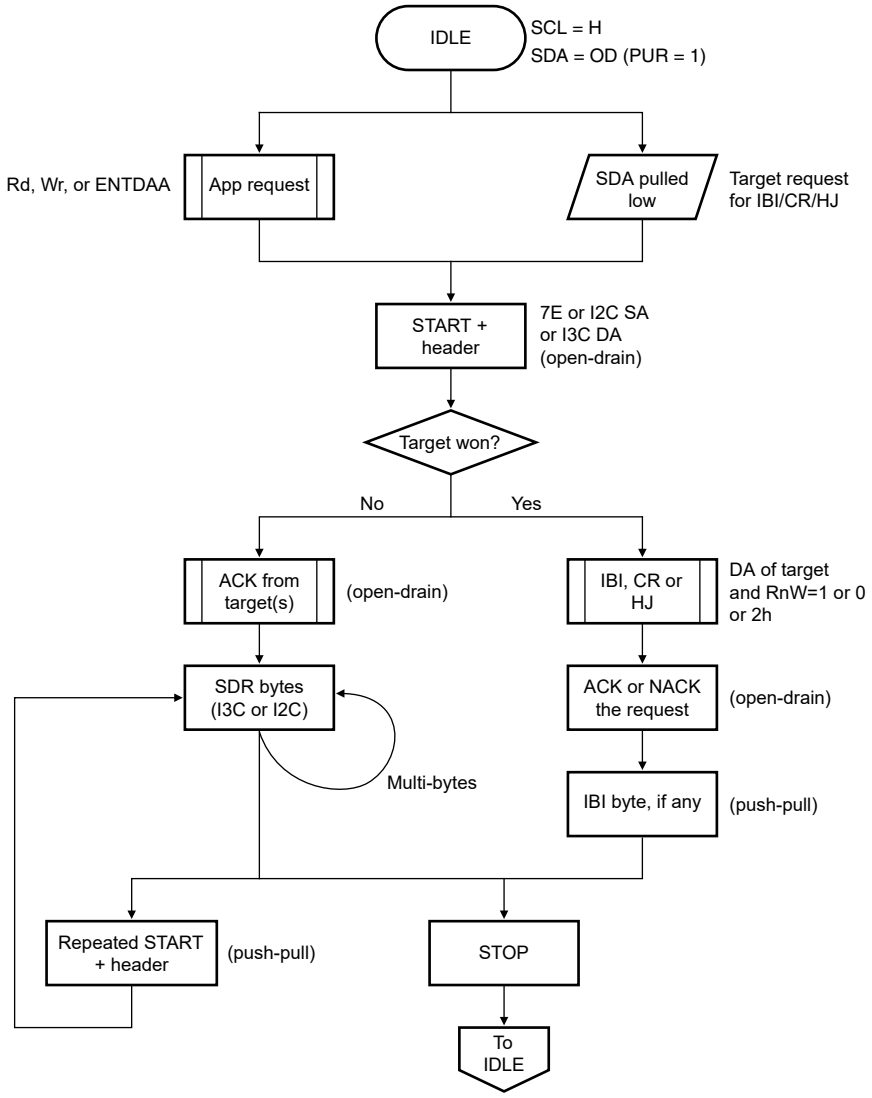


Figure 436. Controller engine flow diagram for SDR (HDR flow progresses from SDR bytes)

72.3.2.10 Target data write from controller

When the address match is valid for the target dynamic address and the type is W (write from controller), the target ACKs the address. It does so unless some condition prevents it, such as a full buffer. On ACK, the Write state is entered. Once ACKed, it waits for each complete data byte. This process occurs in two steps:

1. Eight bits are clocked in and stored in the next buffer location when in the Write state.
2. On the ninth bit:
   - In I3C mode, the W9TH state (in I3C, the ninth data bit written by the controller is the parity of the preceding eight data bits) is used and the parity is checked. The buffer is marked as complete with or without a parity error.
   - In I2C mode, the W9TH state (in I2C, the ninth data bit written by the controller is an ACK by the target) is used and the data is ACKed and stored.

When the address match is valid for the 7Eh broadcast address, the target acknowledges it. After first data completion, it sets the in_ccc broadcast or direct flag (based on bit 7 of the command). If recognized, it parses the command, setting that bit as well. The

recognized commands are for dynamic address work and modes. The CCC and DAA blocks handle the workload afterward. If not supported, then the commands are passed up to the system.

72.3.2.11 Target data write to controller

When the address match is valid for the target dynamic address, and the type is R (read by controller), the target ACKs the address. The target does so unless there is no data waiting for the read. On ACK, the Read state is entered.

After being ACKed, it emits the data byte at a time, with the ninth bit using the R9TH state. In I2C, the ninth data bit from target to controller is an ACK by the controller. In I3C, this bit allows the target to end a read, and allows the controller to abort a read:

- In I3C mode, the T bit is emitted to allow the device to indicate whether this byte is the last. If it is not the last byte, the controller may terminate the read via the T bit.
- In I2C mode, the ninth bit allows the controller to terminate via NACK, or else it allows the read to continue via ACK. On completion of each byte read, the "done" signal is pulsed to get the next byte.

If the read is terminated by the controller unexpectedly (abort in I3C, NACK before END marked), the application is notified.

72.3.3 Clocking

The controller works on the following primary clocks:

- The system clock, which controls access to the memory-mapped registers.
- A functional clock (FCLK), which is used to generate the SCL clock rate on the I2C or I3C bus.

I3C supports adjusting clock frequency and accuracy via Target Time Control Clock (STCCLOCK).

CLK_SLOW is used to check conditions such as bus availability, bus free for IBI, or HJ. The slow clock helps to save power:

- To determine the Bus Available condition for IBI, the target needs a clock to generate the ~1 µs timing. To support this requirement, a slow clock (CLK_SLOW) is provided to save on power.
- I3C supports a counter field, SCONFIG[BAMATCH], to calculate the Bus Available condition. BAMATCH provides the count of the slow clock. This field counts 1 µs or more to allow an IBI to drive SDA low when the controller is free. The I3C module controls the maximum width and maximum values.

I3C supports a timeout when stalled for too long in a frame. This timeout occurs when:

- The transmit FIFO or receive FIFO is not handled and the bus is stuck in the middle of a message.
- No STOP is issued and the bus is between messages.
- IBI manual is used and no decision is made.

72.3.4 Reset

- Global or system reset fed into the module: Everything is reset by this global or system reset. Release is assumed to be synchronized with the system clock. It is not synchronized with SCL or SDA. These signals must not be active when the reset is released; otherwise, I3C enters Hot-Join mode and remains inactive.
- Software reset

72.3.5 Interrupts

This section describes all the interrupts (IRQs) that this module generates. All status interrupts are updated in Controller Status (MSTATUS) and Target Status (SSTATUS).

Supported interrupts occur:

- For pending IBI, CR, or Hot-Join that have been sent by a target.
- When a CCC is received and is handled by the module (SSTATUS[CHANDLED]).

- When an error or warning has occurred, such as data underrun, data overrun, parity error, HDR-DDR, or CRC error.

72.4 External signals

Table 698. External signals

Signal	Description	Direction
SCL	Serial clock	Input or output
SDA	Serial data	Input or output
PUR	Pull up resistance. There is an internal pull-up resistance on SDA, which is controlled by the I3C controller. If the internal pullup is not enough, PUR can be used to control an external pull-up resistance on SDA actively.	Output

72.5 Initialization

72.5.1 Controller configuration

Certain configuration is performed when initializing this module. Controller Configuration (MCONFIG) controls the frequencies, duty cycle, optimizations for performance, and other parameters.

Table 699. Configuration for module initialization

Configuration parameter		Description
MSTENA	Controller enable	Determines whether the peripheral starts in Controller mode (main controller in I3C terms) or starts in Target mode (and switches to Controller mode later).
HKEEP	High-Keeper	Determines how the high-keeper (weak pullup) is implemented, depending on the device capabilities.
PPBAUD	Push-Pull baud	Sets the push-pull frequency as a divider from the FCLK (alternative clock fed to the peripheral). This frequency sets the SCL half-clock period baseline (used at least for the high time of SCL). The formula for SCL in Push-Pull mode using PPBAUD and PPLOW is: SCL frequency (in MHz) = FCLK ÷ ((2 + 2 × PPBAUD) + PPLOW) If PPLOW is 0, then SCL high = SCL low (in ns) = (1 + PPBAUD) × 1000 ÷ FCLK (in MHz) Examples: • If FCLK = 24 MHz, then PPBAUD = 0 yields 12 MHz (42.67 ns per half period) • If FCLK = 50 MHz, then PPBAUD = 1 yields 12.5 MHz (20 + 20 = 40 ns per half period)
PPLOW	Push-Pull low	Changes the duty cycle for Push-Pull. It indicates how many more FCLK cycles to use for low. The formula for SCL in Push-Pull mode using PPBAUD and PPLOW is: SCL frequency (in MHz) = FCLK ÷ ((2 + 2 × PPBAUD) + PPLOW) If PPLOW is a nonzero value, then: • SCL high (in ns) = (1 + PPBAUD) × 1000 ÷ FCLK (in MHz) • SCL low (in ns) = (1 + PPBAUD + PPLOW) × 1000 ÷ FCLK (in MHz) For example, when FCLK is 50 MHz, PPBAUD is 1, and PPLOW is 1, then the periods are:

Table continues on the next page...

Table 699. Configuration for module initialization (continued)

Configuration parameter		Description
		• (1 + 1) × 1000 ÷ 50 = 20 + 20 = 40 ns high • (1 + 1 + 1) × 1000 ÷ 50 = 20 + 20 + 20 = 60 ns low This timing is equivalent to 10 MHz SCL, but this timing maintains the 40 ns high needed; therefore, I2C devices do not see the high periods. **NOTE** The PPLOW value does not have any impact on Open-Drain mode and I2C mode SCL rate calculations.
ODBAUD	Open-Drain baud	Determines the number of PPBAUD periods to make up one I3C Open-Drain half-clock baseline. If ODHPP is 0, the ODBAUD half-clock time period = (ODBAUD + 1) × PPBAUD high period. If ODHPP is 0, the formula for SCL in Open-Drain mode is: SCL (in MHz) = FCLK ÷ (2 + 2 × PPBAUD) × (ODBAUD + 1) Target to get open-drain half-clock period is 200 ns. For example: • If PPBAUD yields 12.5 MHz (40 ns per PPBAUD period), then ODBAUD = 4 can be used to get 200 ns. See ODHPP for details on short high and long low. • If PPBAUD = 1, FCLK = 50 MHz, and ODBAUD = 4, then open-drain SCL = 50 ÷ (2 + 2 × 1) × 5 = 2.5 MHz.
ODHPP	Open-Drain High Push-Pull	Optional field that allows the I3C open drain to be long low and short high. The high period of SCL is the PPBAUD period. This period leaves enough time for the pull-up resistor to pull the SDA high when SCL is low. It is also quick when SCL is high and there are no changes happening. ODBAUD low half-clock time period = (ODBAUD + 1) × PPBAUD high period. ODBAUD high half-clock time period = PPBAUD high period If ODHPP is 1, the formula for SCL in Open-Drain mode is: SCL (in MHz) = FCLK ÷ (1 + PPBAUD) × (ODBAUD + 1) + (1 + PBAUD) For example: • If PPBAUD produces 12.5 MHz (40 ns per PPBAUD period), then ODBAUD = 4 and ODHPP = 1. These settings provide a high period of 40 ns and a low period of 200 ns. • If PPBAUD = 1, FCLK = 50 MHz, and ODBAUD = 4, then open-drain SCL = 50 ÷ (1 + 1) × 5 + (1 + 1) = 4.16 MHz.
I2CBAUD	I2C baud	Indicates the number of ODBAUD periods that are required to communicate with I2C devices (MCTRL[TYPE] = 2 or MWMSG_SDR_CONTROL[I2C] = 1). For example, if ODBAUD gives 200 ns, and the goal is Fm+ (Fast Mode, 1 MHz), then the sum must be 1 µs. I2CBAUD acts differently for odd and even values. For example:

Table continues on the next page...

Table 699. Configuration for module initialization (continued)

Configuration parameter		Description
		• If I2CBAUD = 3, it gives three ODBAUD periods low and two ODBAUD periods high. • If I2CBAUD = 4, it gives four ODBAUD periods for low and four ODBAUD periods for high. • Also, if I2CBAUD = 3, this yields 200 × 3 = 600 ns and 200 × 2 = 400 ns, with the sum 600 + 400 = 1000 ns = 1 µs.
SKEW	Skew	The normal SDA skew from SCL is handled by using the time for the SCL to reach its pad and return. This time is normally 2 ns to 5 ns (or sometimes more). If the skew is too fast, then add more delay. The skew allows specifying the number of FCLKs to insert.

Additional optimizations:

- Use MIBIRULES[MSB0] to obtain faster start header times. If the controller application assigns all I3C dynamic addresses to be less than 40h (it does not have MSB set), you can write 1 to MIBIRULES[MSB0]. When the controller emits 7Eh (broadcast) and a target does not drive the first bit low, the rest of the header can be at push-pull speeds. This speed is two times faster or more, depending on optimizations.
- Auto-emit 7Eh speeds up the frame when used with MIBIRULES[MSB0]. It allows the processor to sleep when the frame starts automatically (in response to a target).

72.5.2 Interrupt service flow

Set up interrupts for Controller Status (MSTATUS) in Controller Interrupt Set (MINTSET).

Set up interrupts for Target Status (SSTATUS) in Target Interrupt Set (SINTSET).

72.6 Application information

72.6.1 I2C configuration to meet timing requirements for Fm and Fm+ modes

I2C Fm and I2C Fm+ modes are supported according to the I3C standard specifications.

Table 700 shows the configuration for Fm mode, where frequency is close to 400 kHz but timing requirement is not met (T_{SU_STA} is not met) because the actual value is 600 ns. Rest of the requirements are met as per the I2C data sheet.

Table 701 shows the configuration for Fm mode, where all timing requirements are met.

Table 700. Configuration for Fm mode with not all timing requirements met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL frequency	Timing requirement status
24 MHz	0	4	11	370 kHz	T_{SU_STA} - 353 ns
48 MHz	1	4	11	370 kHz	T_{SU_STA} - 353 ns
133/134 MHz	5	4	11/12	412 kHz / 382 kHz	T_{SU_STA} - 267 ns and T_{SU_STO} - 529 ns
160 MHz	6	4	10	380 kHz	T_{SU_STA} - 312 ns

Table 701. Configuration for Fm mode with all timing requirements met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL frequency	Timing requirement status
24 MHz	1	2	9	363 kHz	All requirements met as per the I2C data sheet
48 MHz	3	2	9	363 kHz	All requirements met as per the I2C data sheet
133/134 MHz	11	2	9/8	338 kHz / 372 kHz	All requirements met as per the I2C data sheet
160 MHz	13	2	9/8	345 kHz / 422 kHz	All requirements met as per the I2C data sheet

Table 702 shows the configuration for Fm+ mode, where the frequency is close to 1 MHz. In this case, the T_{SU_STA} timing requirement is not met because the actual value is 260 ns. Rest of the requirements are met as per the I2C data sheet.

Table 703 shows the configuration for Fm+ mode, where all timing requirements are met. To meet all the timing specifications, frequency may differ from 1 MHz. Use the configurations listed in Table 703.

Table 702. Configuration for Fm+ mode with not all timing requirements met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL frequency	Timing requirement status
24 MHz	0	4	3	960 kHz	T_{SU_STA} - 228 ns
48 MHz	1	4	3	960 kHz	T_{SU_STA} - 228 ns
133/134 MHz	4	4	3	1.087 MHz	T_{SU_STA} - 156 ns
160 MHz	6	4	3	912 kHz	T_{SU_STA} - 181 ns

Table 703. Configuration for Fm+ mode with all timing requirements met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL frequency	Timing requirement status
24 MHz	0	3	6	749 kHz	All requirements met as per the I2C data sheet
48 MHz	1	3	6	749 kHz	All requirements met as per the I2C data sheet
133/134 MHz	4	3	6	839 kHz	All requirements met as per the I2C data sheet
160 MHz	6	3	6	713 kHz	All requirements met as per the I2C data sheet

NOTE

The controller module provides an option to slow down the clock for standard speed mode to support legacy I2C standard targets as well (although this option may not necessarily meet the exact timing requirement of the standard mode according to the standard I2C timing specification).

For the timing requirements under consideration, see the "I3C timing requirements when communicating with I2C legacy devices" table in the MIPI I3C v1.1.1 specification available on www.mipi.org.

72.7 I3C register descriptions

Writing to a read-only (RO) register, except to Controller Interrupt Mask (MINTMASKED), causes bus errors. This module does not verify whether programmed values in the registers are correct. You must ensure that valid programmed values are written.

However, the peripheral ignores writes to RO registers and returns 0 for reads from WO or nonexistent registers. It requires all 32-bit registers to be read and written as 32-bit and aligned to 32 bits. The peripheral uses the advanced peripheral bus (APB: AMBA 3 APB Protocol Specification v1.0); therefore, the peripheral does not know whether partial read or write operations (less than 32 bits) are used.

72.7.1 I3C memory map

I3C0 base address: 4002_1000h

I3C1 base address: 4002_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Controller Configuration (MCONFIG)	32	RW	0000_0000h
4h	Target Configuration (SCONFIG)	32	RW	0001_0000h
8h	Target Status (SSTATUS)	32	RW	0000_1400h
Ch	Target Control (SCTRL)	32	RW	0000_0000h
10h	Target Interrupt Set (SINTSET)	32	RW	0000_0000h
14h	Target Interrupt Clear (SINTCLR)	32	RW	See section
18h	Target Interrupt Mask (SINTMASKED)	32	R	0000_0000h
1Ch	Target Errors and Warnings (SERRWARN)	32	RW	0000_0000h
20h	Target DMA Control (SDMACTRL)	32	RW	0000_0010h
2Ch	Target Data Control (SDATACTRL)	32	RW	8000_0030h
30h	Target Write Data Byte (SWDATAB)	32	RW	See section
34h	Target Write Data Byte End (SWDATABE)	32	RW	See section
38h	Target Write Data Halfword (SWDATAH)	32	RW	See section
3Ch	Target Write Data Halfword End (SWDATAHE)	32	RW	See section
40h	Target Read Data Byte (SRDATAB)	32	R	0000_0000h
48h	Target Read Data Halfword (SRDATAH)	32	R	See section
54h	Target Write Data Byte (SWDATAB1)	32	RW	0000_0000h
5Ch	Target Capabilities 2 (SCAPABILITIES2)	32	R	0000_0300h
60h	Target Capabilities (SCAPABILITIES)	32	R	E83F_FE78h
64h	Target Dynamic Address (SDYNADDR)	32	RW	0000_0000h
68h	Target Maximum Limits (SMAXLIMITS)	32	RW	0000_0000h
6Ch	Target ID Part Number (SIDPARTNO)	32	RW	3000_0000h
70h	Target ID Extension (SIDEXT)	32	RW	0000_EF00h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
74h	Target Vendor ID (SVENDORID)	32	RW	0000_011Bh
78h	Target Time Control Clock (STCCLOCK)	32	RW	0000_0214h
7Ch	Target Message Map Address (SMSGMAPADDR)	32	R	0000_0000h
84h	Controller Control (MCTRL)	32	RW	0000_0000h
88h	Controller Status (MSTATUS)	32	RW	0000_1000h
8Ch	Controller In-band Interrupt Registry and Rules (MIBIRULES)	32	RW	0000_0000h
90h	Controller Interrupt Set (MINTSET)	32	RW	0000_0000h
94h	Controller Interrupt Clear (MINTCLR)	32	RW	See section
98h	Controller Interrupt Mask (MINTMASKED)	32	R	0000_0000h
9Ch	Controller Errors and Warnings (MERRWARN)	32	RW	0000_0000h
A0h	Controller DMA Control (MDMACTRL)	32	RW	0000_0010h
ACh	Controller Data Control (MDATACTRL)	32	RW	8000_0030h
B0h	Controller Write Data Byte (MWDATAB)	32	RW	See section
B4h	Controller Write Data Byte End (MWDATABE)	32	RW	See section
B8h	Controller Write Data Halfword (MWDATAH)	32	RW	See section
BCh	Controller Write Data Halfword End (MWDATAHE)	32	RW	See section
C0h	Controller Read Data Byte (MRDATAB)	32	R	0000_0000h
C8h	Controller Read Data Halfword (MRDATAH)	32	R	0000_0000h
CCh	Controller Write Byte Data 1 (to Bus) (MWDATAB1)	32	RW	0000_0000h
D0h	Controller Write Message Control in SDR mode (MWMSG_SDR_CONTROL)	32	RW	See section
D0h	Controller Write Message Data in SDR mode (MWMSG_SDR_DATA)	32	RW	0000_0000h
D4h	Controller Read Message in SDR mode (MRMSG_SDR)	32	R	0000_0000h
D8h	Controller Write Message in DDR mode: First Control Word (MWMSG_DDR_CONTROL)	32	RW	See section
D8h	Controller Write Message in DDR Mode Control 2 (MWMSG_DDR_CONTROL2)	32	RW	See section
D8h	Controller Write Message Data in DDR mode (MWMSG_DDR_DATA)	32	RW	0000_0000h
DCh	Controller Read Message in DDR mode (MRMSG_DDR)	32	R	0000_0000h
E4h	Controller Dynamic Address (MDYNADDR)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
11Ch	Map Feature Control 0 (SMAPCTRL0)	32	R	0000_0000h
140h	Extended IBI Data 1 (IBIEXT1)	32	RW	0000_0070h
144h	Extended IBI Data 2 (IBIEXT2)	32	RW	0000_0000h
FFCh	Target Module ID (SID)	32	R	EDCB_0100h

72.7.2 Controller Configuration (MCONFIG)

Offset

Register	Offset
MCONFIG	0h

Function

Controls all controller states when the controller operation is enabled. You must not change this register during an active transaction.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	I2CBAUD				SKEW			ODHPP	ODBAUD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PPLOW				PPBAUD				Reserved	ODSTOP	HKEEP		DISTO	0	MSTENA	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 I2CBAUD	I2C Baud Rate Specifies the I2C low and high times in ODBAUD counts: • I2CBAUD >> 1 is the main count load, and it is count - 1. So, I2CBAUD >> 1: I2CBAUD = 0 for one ODBAUD beat and I2CBAUD = 1 for two ODBAUD beats.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• If I2CBAUD[28] is 1, then the low time has one extra ODBAUD beat. The I2CBAUD field is normally 3, where ODBAUD gives 200 ns. For I2CBAUD >> 1, I2CBAUD = 1, which means two ODBAUD beats. To meet the requirements for Fast-mode Plus (Fm+), (2 + 1) × 200 = 600 ns low, with 2 × 200 = 400 ns high for 1 µs period. For Fast mode (Fm), I2CBAUD is normally 11 (giving 2.6 µs) or 6 (giving 2.4 µs).
27-25 SKEW	Skew Specifies the number of FCLK counts for an SDA change after SCL for I3C push-pull. This skew is in addition to the roundtrip of the SCL line from the pad back to the module (6 ns or more): • SKEW is normally not needed, so assign SKEW = 0. • SKEW is only used if SDA is not naturally skewing from an SCL change. • I2C automatically skews SDA (but not PUR) to match I2C rules. Use the following values: • In I2C Controller mode (MCONFIG[MSTENA] = 11b) — SKEW > 0 when clock stretching feature is not required. — SKEW = 0 when clock stretching feature is required. • In I3C Controller mode, use SKEW = 0.
24 ODHPP	Open Drain High Push-Pull Enables ODHPP. If you write 0 to this field, open-drain SCL high half-clock period is the same as the open-drain low SCL half period. If you write 1 to this field, open-drain high SCL half-lock period is one PPBAUD count for I3C messages. This setting is faster and works for I3C devices. Any legacy I2C devices on the bus do not see the SCL high level at all (less than the spike filter period). For open-drain timing, check upon the first 7Eh broadcast allowing I3C peripherals acting as I2C legacy devices to turn off their I2C 50-ns spike filters. See Sending a CCC to I3C targets for more information. 0b - Disable 1b - Enable
23-16 ODBAUD	Open Drain Baud Rate Specifies the open drain baud rate. ODBAUD is configured in terms of number of PPBAUD counts - 1. This field must not be 0; this setting is not the same as push-pull. See Controller configuration for more information. The open-drain baud rate is usually 200 ns (see I2CBAUD for I2C counts). When used with MCONFIG[ODHPP], this setting produces 250 ns per clock in I3C. In this case, if MCONFIG[PPBAUD] provides 12 MHz, then one PPBAUD count is half of 12 MHz, or 41.67 ns. To obtain a rate around 200 ns, use a value of 5 - 1 = 4 for ODBAUD.
15-12 PPLOW	Push-Pull Low Acts as an adder for push-pull low to create a duty cycle with a longer low period, with up to 15 more FCLK cycles low than high. PPLOW = 0 produces a 50/50 duty cycle.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 PPBAUD	Push-Pull Baud Rate Specifies the push-pull baud rate. The number of FCLK counts makes each push-pull low and normally high period. PPBAUD = 0 when run at 1/2 input FCLK speed. For example, a 24 MHz FCLK produces a 12 MHz SCL because each FCLK is SCL low or SCL high. MCONFIG[PPLOW] has an effect on the duty cycle. For example, 24 MHz with 50/50 duty cycle is 12 MHz. However, when PPLOW adds three more low beats, the push-pull baud rate becomes 4.8 MHz (from 24 MHz or 5 beats).
7 —	Reserved
6 ODSTOP	Open Drain Stop Enables open-drain stop. If you write 0 to this field, open-drain stop is disabled. ODSTOP must be disabled when sending an HDR exit pattern. If you write 1 to this field, open-drain stop is enabled. STOP condition is emitted at open-drain speeds even for I3C messages. In legacy devices, this feature can ensure that the legacy devices see the STOP condition. 0b - Disable 1b - Enable
5-4 HKEEP	High-Keeper Indicates how High-Keeper is supported. If this field is 0b, use pull-up resistor (PUR). No separate pin_HK_SDA or pin_HK_SCL is used. Only pin_PUR_oena is used for SDA. The pin_SCL_oena pin is held high in this mode, SCL clock is push-pull, and pin_SCL_out toggles, which is the actual clock. If this field is 1b, High-Keeper controls are in place; pin_HK_SDA or pin_HK_SCL (High-Keeper) controls are used. SCL may use an HK or that signal (pin_HK_SCL) may only OR into pin_SCL_oena. The pin_HK_SDA pin is used as well for SDA high keeper, along with pin_PUR_oena. If this field is 10b, the functionality is passive on SDA, can Hi-Z (high-impedance) for bus free (Idle) and hold. The pins, pin_HK_SDA and pin_HK_SCL, are not used; only a combination of SDA_oena and pin_PUR_oena are used. If this field is 11b, the functionality is passive on SDA and SCL, can Hi-Z (high-impedance) both for bus free (Idle), and can Hi-Z SDA for hold. This is for I2C Clock Stretching mode, where both SCL and SDA are open drain. Use HKEEP = 2/3 for I2C modes. Use HKEEP = 0/1 for I3C modes. **NOTE** Your chip may not support any or all High-Keeper methods. You must check the availability of High-Keeper methods. 00b - None

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - WIRED_IN 10b - PASSIVE_SDA 11b - PASSIVE_ON_SDA_SCL
3 DISTO	Disable Timeout Disables the timeout that produces application errors. If the controller is left in a state other than Stopped for more than 100 µs (because 10 kHz is the slowest allowed I3C speed), the timeout sends a MERRWARN interrupt. To prevent the MERRWARN interrupt during development or testing, write 1 to DISTO to disable the timeout. In systems that support timeouts, timeout is disabled automatically during debug. 0b - Enabled 1b - Disabled, if configured
2 —	Reserved
1-0 MSTENA	Controller Enable Indicates whether the controller is enabled and the states it can use. If this field is 0b, the controller is disabled. The module can only use Target mode. If this field is 1b, the controller is enabled. When used upon start-up, this module is the main controller by default. I3C controls the bus unless the controller is handed off. When this happens, MSTENA must become 2 so that it has the capability to become the controller again. Performing the handoff means emitting the GETACCMST CCC command. If the command is accepted, I3C emits a STOP condition and sets this field to 2 (or 0). If this field is 10b, I3C is controller-capable, but is operating as a target now. When used from the start, I3C starts as a target, but is prepared to switch to Controller mode. To switch to this mode, the target emits a controller request (CR) or receives a GETACCMST CCC command and accepts it (to switch on the STOP condition). Legacy I2C—if this field is 11b, I2C Controller mode uses an open-drain clock (to allow clock stretching) and relies on passive pull-up resistors (PURs) on both SCL and SDA. This mode uses only I2CBAUD. START requests can only be made for I2C type, and I3C requests are not accepted. If there are only I2C targets on the bus and no clock stretching, use type 1 instead. Type 1 has a push-pull SCL clock, which is faster and uses less power. 00b - CONTROLLER_OFF 01b - CONTROLLER_ON 10b - CONTROLLER_CAPABLE 11b - I2C_CONTROLLER_MODE

72.7.3 Target Configuration (SCONFIG)

Offset

Register	Offset
SCONFIG	4h

Function

Contains fields that must be configured before the module is activated.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SADDR							0	BAMATCH							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0		OFFLINE	IDRAND	0	0		DDROK	S0IGNORE	MATCHSS	NACK	SLVENA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 SADDR	Static Address Sets the I2C 7-bit static address, which otherwise must be 0.
24 —	Reserved
23-16 BAMATCH	Bus Available Match Specifies the bus available condition match value for the current slow clock. BAMATCH provides the count of the slow clock to count out 1 µs (or more) to allow an IBI to drive SDA low when the controller is not doing so. I3C controls the maximum width and maximum values.
15-12 —	Reserved
11-10 —	Reserved
9	Offline

Table continues on the next page...

Table continued from the previous page...

Field	Function
OFFLINE	Enables wait to ensure that the bus is not in HDR mode. If this field is 1 when SCONFIG[SLVENA] is 1, then I3C waits for either 60 µs of bus quiet or an HDR exit pattern. This waiting ensures that the bus is not in HDR mode, and so can safely monitor the next activity in Single Data Rate (SDR) mode. 0b - Disable 1b - Enable
8 IDRAND	ID random Specifies whether SIDPARTNO[PARTNO] is a random value. 0b - Part number and an instance 1b - Random value
7 —	Reserved
6-5 —	Reserved
4 DDROK	Double Data Rate OK Allows HDR-DDR messaging. If DDROK = 1, write 1 to the corresponding SIDEXT[BCR] field to indicate that high data rate (HDR) is available, and configure the corresponding HDRCAP HDR-DDR field to allow using DDR mode. **NOTE** DDROK must be 1 before the target can connect to the I3C bus. The target peripheral indicates to the controller whether the target can support the feature during dynamic address assignment. 0b - Do not allow HDR-DDR messaging 1b - Allow HDR-DDR messaging
3 S0IGNORE	Ignore TE0 or TE1 Errors Ignores TE0 or TE1 errors. If you write 1 to this field, the target does not detect TE0 or TE1 errors, so it does not lock up waiting on an exit pattern. You must not use this setting when the bus does not use HDR mode. 0b - Do not ignore TE0 or TE1 errors 1b - Ignore TE0 or TE1 errors
2 MATCHSS	Match Start or Stop Enables match START or STOP condition. If you write 1 to this field, SINTSET[START] and SINTSET[STOP] become 1 only when SSTATUS[MATCHED] is 1. This setting allows the START and STOP fields to be used to detect the end of a message to or from this target.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
1 NACK	Not Acknowledge Controls the ACK or NACK capability. If you write 1 to this field, the target rejects all requests, except for a CCC broadcast. NACK = 1 must be used with caution because the controller may decide that the target is missing, if NACK is overused. 0b - Always disable NACK mode 1b - Always enable NACK mode (works normally)
0 SLVENA	Target Enable Enables the target. If you write 0 to this field, the target ignores the I2C or I3C bus. If you write 1 to this field, the target can operate on the I2C or I3C bus. You must not write 1 to this field before registers such as Target Configuration (SCONFIG), Target ID Part Number (SIDPARTNO), Target ID Extension (SIDEXT), and others are configured because these registers affect the data to and from the controller. Target enable is configured just once before the bus comes up. If target enable is used at other times, SCAPABILITIES[IBI_MR_HJ] must be 1 before writing 1 to SLVENA so that the device does not see a START or STOP condition incorrectly. 0b - Disable 1b - Enable

72.7.4 Target Status (SSTATUS)

Offset

Register	Offset
SSTATUS	8h

Function

Indicates the sticky status for interrupts, states, and modes related to the I3C bus. Not all fields of the register are used if the module only acts as a target. The fields are divided into current activity, interrupt maskable actions, and then states and modes on the bus.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TIMECTRL		ACTSTATE		HJDIS	0	MRDIS	IBIDIS	0		EVDET		0	EVENT	CHANDLED	HDRMATCH
W														W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERRWARN	CCC	DACHG	TXNOTFU...	RX_PEND	STOP	MATCHED	START	0	STHDR	STDAA	STREQWR	STREQRD	STCCCH	STMSG	STNOTST...
W		W1C	W1C			W1C	W1C	W1C								
Reset	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 TIMECTRL	Time Control Indicates whether time control is enabled, and in which mode. 00b - NO_TIME_CONTROL (no time control is enabled) 01b - SYNC_MODE (Synchronous mode is enabled) 10b - ASYNC_MODE (Asynchronous standard mode (0 or 1) is enabled) 11b - BOTHSYNCASYNC (both Synchronous and Asynchronous modes are enabled)
29-28 ACTSTATE	Activity State from Common Command Codes (CCC) Indicates the activity state from CCC. 00b - NO_LATENCY (normal bus operations) 01b - LATENCY_1MS (1 ms of latency) 10b - LATENCY_100MS (100 ms of latency) 11b - LATENCY_10S (10 seconds of latency)
27 HJDIS	Hot-Join Disabled Indicates whether hot-join is disabled. When hot-join is disabled, CTRL requests are not responded to. 0b - Enabled 1b - Disabled
26 —	Reserved
25 MRDIS	Controller Requests Disable Indicates whether controller requests are disabled. When controller requests are disabled, CTRL requests are not responded to.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enabled 1b - Disabled
24 IBIDIS	In-Band Interrupts Disable Indicates whether in-band interrupts are disabled. When in-band interrupts are disabled, CTRL requests are not responded to. 0b - Enabled 1b - Disabled
23-22 —	Reserved
21-20 EVDET	Event Details Indicates current details of the last (EVENT = 1) or pending event. 00b - NONE (no event or no pending event) 01b - NO_REQUEST (request is not sent yet; either there is no START condition yet, or is waiting for Bus-Available or Bus-Idle (HJ)) 10b - NACKed (not acknowledged, request sent and rejected); I3C tries again 11b - ACKed (acknowledged; request sent and accepted), so done (unless the time control data is still being sent)
19 —	Reserved
18 EVENT	Event Indicates, for a target, whether a pending in-band interrupt (IBI), controller request (CR), or hot-join (HJ) has been sent as requested. See the upper status register fields for details. This field becomes 1 only when acknowledged by a controller. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No event occurred 1b - IBI, CR, or HJ occurred When writing 0b - No effect 1b - Clear the flag
17 CHANDLED	Common Command Code Handled

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates whether an HDRMATCH CCC is being handled by I3C. This field is for notification only, but it may result in updates to this register. **NOTE** This field behaves differently for register reads and writes. When reading 0b - CCC handling not in progress 1b - CCC handling in progress When writing 0b - No effect 1b - Clear the flag
16 HDRMATCH	High Data Rate Command Match Indicates whether the HDR command matched the I3C dynamic address of this device. The HDR command is available as the first byte, with RXPEND = 1. The MSB of the command byte indicates whether it is a read or a write command. If the HDR command is a read, and there are to-bus bytes waiting, then the command is ACKed and the data is sent back. Otherwise, the HDR command is NACKed. **NOTE** When this field is 1, you must check SSTATUS[ERRWARN] because the HPAR error may occur after signaling this HDR command. The parity occurs after the destination address and command. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Did not match 1b - Matched the I3C dynamic address When writing 0b - No effect 1b - Clear the flag
15 ERRWARN	Error Warning Indicates that an error or warning has occurred, such as data underrun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition (see Target Errors and Warnings (SERRWARN) for more information).
14 CCC	Common Command Code Indicates whether a CCC has been received, and is not handled by I3C. There are two types of common command codes: • Broadcast CCC, which corresponds with RXPEND, and the first byte is the CCC (command).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• Direct CCC, which may never be directed to this device. If the direct CCC is directed to this device, then TXSEND or RXPEND are triggered, and RXPEND contains the command. **NOTE** This field behaves differently for register reads and writes. When reading 0b - CCC not received 1b - CCC received When writing 0b - No effect 1b - Clear the flag
13 DACHG	Dynamic Address Change Indicates an occurrence of dynamic address (DA) change. Actual DA can be seen in the DYNADDR register. If this field is 1, DA change is detected. The target DA has been assigned, reassigned, or reset (lost) and is now in the state of being valid or none. This field is also used when the MAP auto feature is configured, changing one or more MAP items. See DYNADDR and MAPCTRL*n* for more information. DYNAADDR for the main DA (0) indicates whether the last change was because of Auto-MAP. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No DA change detected 1b - DA change detected When writing 0b - No effect 1b - Clear the flag
12 TXNOTFULL	Transmit Buffer Not Full Indicates that the transmit buffer is not full. If DMA is enabled for transmitting, then it is also signaled to provide more data. To-bus buffer or FIFO can accept more data to be transmitted. For all but external FIFOs, this process uses SDATACTRL[TXTRIG], which defaults to "not full". 0b - Transmit buffer full 1b - Transmit buffer not full
11 RX_PEND	Received Message Pending Indicates whether a received message is pending. The field indicates when a message from the controller that is not being handled by I3C (not a CCC message) is received. I3C processes such messages internally. For all but external FIFOs, this process uses SDATACTRL[RXTRIG], which

Table continues on the next page...

Table continued from the previous page...

Field	Function
	defaults to "not empty". If DMA is enabled for receiving, then DMA is signaled as well. This field automatically becomes 0 if data is read (from FIFO and non-FIFO sources). 0b - No received message pending 1b - Received message pending
10 STOP	Stop Indicates whether the Stopped state is detected. The STNOTSTOP state also indicates when the module is in Stop mode. A fast STOP and START combination may not trigger the Stop status. In that case, the Start status is always set. If this field is 1, it indicates that the Stop state was present on the bus since the bus was last cleared. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No Stopped state detected 1b - Stopped state detected When writing 0b - No effect 1b - Clear the flag
9 MATCHED	Matched Indicates whether an incoming header matched the I3C dynamic or I2C static address (if any) of this device since the bus was last cleared. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Header not matched 1b - Header matched When writing 0b - No effect 1b - Clear the flag
8 START	Start Indicates whether a START or Repeated START was detected after this flag was last cleared. This flag is not usually needed, but can be used for wake events.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
7 —	Reserved
6 STHDR	Status High Data Rate Indicates whether the I3C bus is in HDR-DDR mode, regardless of whether HDR mode is supported by this module, and regardless of whether the message is intended for this module or some other module. 0b - I3C bus not in HDR-DDR mode 1b - I3C bus in HDR-DDR mode
5 STDAA	Status Dynamic Address Assignment Indicates whether the I3C bus is in Enter Dynamic Address Assignment (ENTDAA) mode, regardless of whether this bus target has a dynamic address. 0b - Not in ENTDAA mode 1b - In ENTDAA mode
4 STREQWR	Status Request Write Indicates whether the REQ in process is SDR write data from the controller to this bus target (or all targets), but not in ENTDAA mode. See status high data rate (STHDR) for double data rate (DDR) handling. 0b - Not an SDR write 1b - SDR write data from the controller, but not in ENTDAA mode
3 STREQRD	Status Request Read Indicates whether the REQ in process is an SDR read from this target. See status high data rate (STHDR) for double data rate (DDR) handling. 0b - Not an SDR read 1b - SDR read from this target or an IBI is being pushed out
2	Status Common Command Code Handler

Table continues on the next page...

Table continued from the previous page...

Field	Function
STCCCH	Indicates whether a CCC message is being handled automatically. 0b - No CCC message handled 1b - Handled automatically
1 STMSG	Status Message Indicates whether the bus target is busy (listening to the bus traffic or responding). If STNOSTOP = 1, STMSG is 0 when a nonmatching address is seen, until the next Repeated START or STOP condition occurs. 0b - Idle 1b - Busy
0 STNOTSTOP	Status not Stop Indicates the status of the bus. If this field is 0, I3C is in a STOP condition. If this field is 1, the bus is busy (has activity). Other fields of this register may also be set when busy. STNOTSTOP can also become 1 after a TE0 or TE1 error, when I3C is waiting for an exit pattern. 0b - In STOP condition 1b - Busy

72.7.5 Target Control (SCTRL)

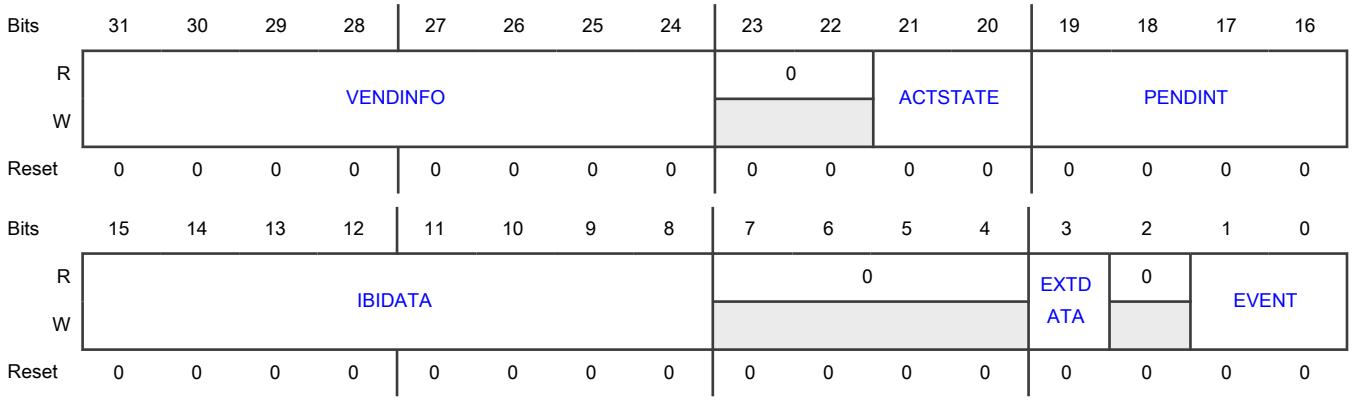
Offset

Register	Offset
SCTRL	Ch

Function

Contains controls for the active use of the I3C bus (for example, event generation such as interrupts to the controller). Only if I3C is configured to support various special operations for the target, this register is used to activate those operations. These events include IBI and GETSTATUS fields (except the protocol error, which is automatically set).

Diagram



Fields

Field	Function
31-24 VENDINFO	Vendor Information Controls vendor information that the GETSTATUS CCC returns. You must set this field to the Vendor Reserved field that the GETSTATUS CCC returns. The application must maintain the vendor information because the controller reads this field. If this field is not configured, then the GETSTATUS field always returns 0.
23-22 —	Reserved
21-20 ACTSTATE	Activity State of Target Controls the activity state of the target. You must set this field to the activity state of the target that the GETSTATUS CCC command returns as activity mode. The application must maintain the activity state because the controller reads this field. If you do not configure the activity state, then the GETSTATUS command always returns 0.
19-16 PENDINT	Pending Interrupt Specifies whether an IBI interrupt is pending. You must set this field to the pending interrupt that the GETSTATUS CCC command returns. The application must maintain the pending interrupt because the controller reads this field. If PENDINT = 0: • The GETSTATUS command returns 1 if an IBI interrupt is pending. • The GETSTATUS field returns 0 if an IBI interrupt is not pending.
15-8 IBIDATA	In-Band Interrupt Data Controls the data byte accompanying the IBI, if the module is enabled for IBI. If SCTRL[IBIDATA] is enabled, then IBI is required.
7-4 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 EXTDATA	Extended Data Enables extended data. After IBIDATA is emitted, extended data is acquired from IBIEXT1 and IBIEXT2, if configured. If extended data is used with time control, the data follows the time information. See IBIEXT1[MAX] for more information. 0b - Disable 1b - Enable
2 —	Reserved
1-0 EVENT	Event Requests an event. If this field is 0b, it indicates Normal mode, in which if this field becomes 0 from a nonzero value and event processing has not yet started, the event processing is canceled. However, event processing is not canceled if it has already started. If this field is 1b, an IBI is pushed onto the I3C bus. If there is data associated with the IBI, the data is read from SCTRL[IBIDATA]. If time control is enabled, this data includes any time-control-related bytes. Additionally, SCTRL[IBIDATA][7] becomes 1 automatically (as is required for time control). The IBI interrupt occurs after the first (mandatory) IBIDATA, if any. If this field is 10b, a controller request is started; the meaning depends on SIDEXT[BCR] configured in I3C. If this field is 11b, a hot-join (HJ) request is started, which is used when the device is powered on after the I3C bus is already powered up. It is also used when the device is connected using hot-insertion methods (the device is powered up when it is physically inserted in the powered-up I3C bus). The HJ waits for Bus Idle, and SCTRL[EVENT] = HOT_JOIN_REQUEST must be set before the target enable (SCONFIG[SLVENA]). If this field is a nonzero value, it requests an event. After the request, SSTATUS[EVENT] and SSTATUS[EVDET] show the status as it progresses. After completion, this field automatically returns to 0. After this field becomes a nonzero value, you can write only 0 it (to cancel) until the event processing is finished. 00b - NORMAL_MODE 01b - IBI 10b - CONTROLLER_REQUEST 11b - HOT_JOIN_REQUEST

72.7.6 Target Interrupt Set (SINTSET)

Offset

Register	Offset
SINTSET	10h

Function

Sets interrupt enables for select Target Status (SSTATUS) fields. Reading this register (SINTSET) returns the status of the interrupt enable:

- To activate an interrupt enable, write 1 to its corresponding field in this register (SINTSET).
- To disable an interrupt, write 1 to its corresponding field in Target Interrupt Clear (SINTCLR). Writing 0 to the interrupt enable in this register (SINTSET) does not disable the interrupt.

The Interrupt registers allow the masking of interrupt sources. They also allow the checking of which interrupts are activated. The normal method is to enable an interrupt, and then after the interrupt occurs, clear the interrupt either by writing to Target Status (SSTATUS) or by performing an action on the corresponding data register. The interrupt is level-held, meaning the interrupt stays set until the cause is cleared by some method. The module prevents races; if a new event occurs, that new event is not lost:

- SINTSET sets interrupt enables for Target Status (SSTATUS) fields. Reading the SINTSET register returns the status of the interrupt enables.
- SINTCLR clears interrupt enables for the SSTATUS fields.
- SINTMASKED returns the value of the SSTATUS fields ANDed with their interrupt enables.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												0	EVENT	CHANDLED	DDRMATC...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERRWARN	CCC	DACHG	TXSEND	RXPEND	STOP	MATCHED	START	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19 —	Reserved
18 EVENT	Event Interrupt Enable Enables event interrupts. This field indicates, for a target, whether a pending in-band interrupt (IBI), controller request (CR), or hot-join (HJ) has been sent as requested. The field is configured to support events. See SSTATUS[EVDET] for more information.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
17 CHANDLED	Common Command Code (CCC) Interrupt Enable Enables CCC interrupts. This field specifies whether CCC is being handled by I3C. Target Status (SSTATUS) shows new results. You can use this field to track when activity states and masks on events (for example, IBIs) occur. The field is used for CCCs that are enabled. 0b - Disable 1b - Enable
16 DDRMATCHED	Double Data Rate Interrupt Enable Enables DDR match for read or write command. This field indicates when DDR matches for read or write command, and the field is used only if HDR is enabled. 0b - Disable 1b - Enable
15 ERRWARN	Error or Warning Interrupt Enable Enables error or warning interrupts. This field indicates whether an error or warning has occurred, such as data underrun, data overrun, parity error, or HDR-DDR CRC error, or any other error or warning condition. See Target Errors and Warnings (SERRWARN) for cause of error. Only available for errors related to configured features. 0b - Disable 1b - Enable
14 CCC	Common Command Code (CCC) Interrupt Enable Enables CCC interrupts. This field indicates whether a CCC has been received, and is not handled by I3C. For CCCs that the block does not handle, RXPEND also interrupts, and SSTATUS[STREQRD] indicates that it is a CCC sending a read request. 0b - Disable 1b - Enable
13 DACHG	Dynamic Address Change Interrupt Enable Enables dynamic address change interrupts: interrupt on dynamic address defined (SETDASA or ENTDAA) or lost (RSTDAA). See Controller Dynamic Address (MDYNADDR) for more information.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable 1b - Enable
12 TXSEND	Transmit Interrupt Enable Enables transmit interrupts. This field indicates whether to-bus buffer or FIFO can accept more data to be transmitted. The corresponding interrupt occurs when the controller requests data to read from this target. This interrupt occurs on the first request (header) as well as when ready for more data. The corresponding interrupt occurs when the controller requests data to be read from this target. If this interrupt is for a FIFO, it triggers on the transmit emptiness trigger. If this interrupt is for DMA, then it indicates message end (DMA end or termination). 0b - Disable 1b - Enable
11 RXPEND	Receive Interrupt Enable Enables receive interrupts. This field specifies when a message from the controller that is not being handled by the module is received. For example, where data is consumed by hardware directly when it goes into the FIFO (excludes CCCs being handled automatically). If this interrupt is for a FIFO, it indicates a receive fullness trigger. If this interrupt is for DMA, then it indicates message end. 0b - Disable 1b - Enable
10 STOP	Stop Interrupt Enable Enables stop interrupts. This field detects the Stopped state present on the bus since the bus was last cleared. Use SINTSET[START] as the preferred interrupt when needed. This interrupt may not trigger for a quick STOP and START combination because it relates to the state of being stopped. 0b - Disable 1b - Enable
9 MATCHED	Match Interrupt Enable Enables match interrupts. Incoming header matched the I3C dynamic or I2C static address of this device since the bus was last cleared. If configured and if no dynamic address is set, this interrupt is also for matching a header on an I2C static address. See Controller Dynamic Address (MDYNADDR) for related information. 0b - Disable 1b - Enable
8	Start Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
START	Enables start interrupts. A START or Repeated START (such as wakeup) is seen after this field last became 0. See SINTSET[STOP] for related information. 0b - Disable 1b - Enable
7-0 —	Reserved

72.7.7 Target Interrupt Clear (SINTCLR)

Offset

Register	Offset
SINTCLR	14h

Function

Clears interrupt enables for select Target Status (SSTATUS) fields. To clear an interrupt enable, write 1 to the corresponding field in this register (SINTCLR). Writing 0 has no effect.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												0	EVENT	CHANDLED	DDRMATC...
W														W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERRWARN	CCC	DACHG	TXSEND	RXPEND	STOP	MATCHED	START	0							
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C								
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 —	Reserved
18 EVENT	EVENT Interrupt Enable Clear
17 CHANDLED	CHANDLED Interrupt Enable Clear
16 DDRMATCHED	DDRMATCHED Interrupt Enable Clear
15 ERRWARN	ERRWARN Interrupt Enable Clear
14 CCC	CCC Interrupt Enable Clear
13 DACHG	DACHG Interrupt Enable Clear
12 TXSEND	TXSEND Interrupt Enable Clear
11 RXPEND	RXPEND Interrupt Enable Clear
10 STOP	STOP Interrupt Enable Clear
9 MATCHED	Matched Interrupt Enable Clear
8 START	START Interrupt Enable Clear
7-0 —	Reserved

72.7.8 Target Interrupt Mask (SINTMASKED)

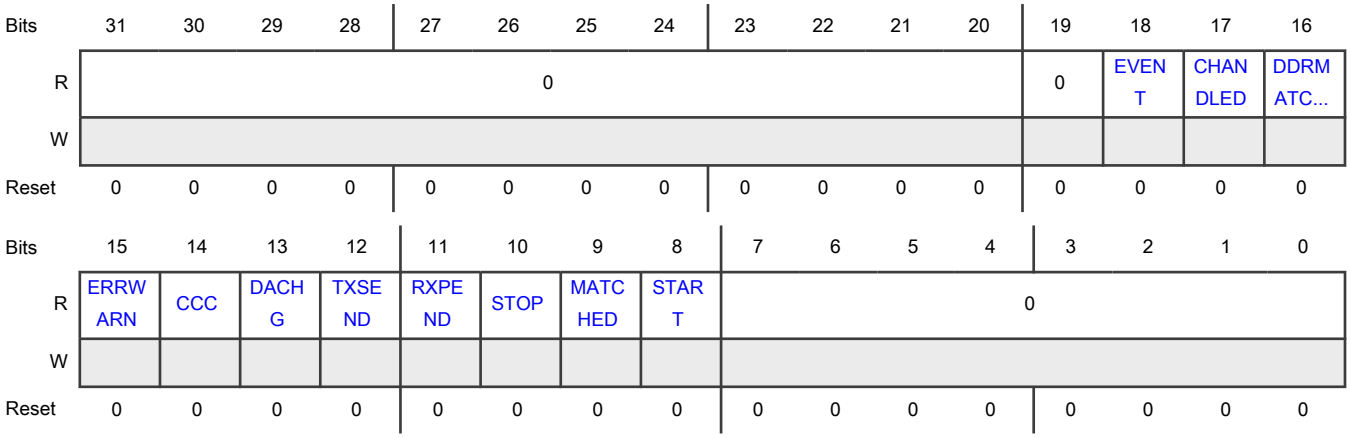
Offset

Register	Offset
SINTMASKED	18h

Function

Returns the status of enabled interrupts (the value of Target Status (SSTATUS) ANDed with the value of Target Interrupt Set (SINTSET)).

Diagram



Fields

Field	Function
31-20 —	Reserved
19 —	Reserved
18 EVENT	EVENT Interrupt Mask
17 CHANDLED	CHANDLED Interrupt Mask
16 DDRMATCHED	DDRMATCHED Interrupt Mask
15	ERRWARN Interrupt Mask

Table continues on the next page...

Table continued from the previous page...

Field	Function
ERRWARN	
14 CCC	CCC Interrupt Mask
13 DACHG	DACHG Interrupt Mask
12 TXSEND	TXSEND Interrupt Mask
11 RXPEND	RXPEND Interrupt Mask
10 STOP	STOP Interrupt Mask
9 MATCHED	MATCHED Interrupt Mask
8 START	START Interrupt Mask
7-0 —	Reserved

72.7.9 Target Errors and Warnings (SERRWARN)

Offset

Register	Offset
SERRWARN	1Ch

Function

Contains errors and warnings from I3C and I2C protocols. This includes internal issues such as overrun and underrun, detected errors and conditions like parity errors, CRC errors, and read terminations by the controller. See SSTATUS[ERRWARN] and SINTSET[ERRWARN] for related information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										0		0		OWRITE	OREAD
W															W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		0	0	S0S1	HCRC	HPAR	SPAR	0			INVSTART	TERM	URUNNACK	URUN	ORUN
W					W1C	W1C	W1C	W1C				W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-22 —	Reserved
21 —	Reserved
20-18 —	Reserved
17 OWRITE	Over-Write Error Indicates that Target Write Data Byte (SWDATAB) or Target Write Data Byte End (SWDATABE) was written to when full. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No overwrite error 1b - Overwrite error When writing 0b - No effect 1b - Clear the flag
16 OREAD	Over-Read Error Indicates that Target Read Data Byte (SRDATAB) was read for more bytes than were available, by the application. This field also indicates over-read errors for Target Read Data Halfword (SRDATAH), if it is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - No over-read error 1b - Over-read error When writing 0b - No effect 1b - Clear the flag
15-14 —	Reserved
13 —	Reserved
12 —	Reserved
11 S0S1	TE0 or TE1 Error Indicates whether a TE0 or TE1 error occurred and the target is locked and waiting for an HDR exit pattern. Writing 1 to S0S1 causes I3C to release the lock, but this method must be used with great care. S0S1 becomes 0 automatically when an exit pattern is detected, so writing 1 to S0S1 must be used under controlled circumstances to avoid problems. Before starting to operate normally after this error, I3C waits for a START (or Repeated START) or STOP state. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No TE0 or TE1 error occurred 1b - TE0 or TE1 error occurred When writing 0b - No effect 1b - Clear the flag
10 HCRC	HDR-DDR CRC Error Indicates whether an HDR-DDR CRC error occurred on a message from the controller. Error reasons include an HDR restart and an exit being issued before an HDR-DDR message from the controller has finished. This error calls into question the data from the entire DDR command frame.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - No HDR-DDR CRC error occurred 1b - HDR-DDR CRC error occurred When writing 0b - No effect 1b - Clear the flag
9 HPAR	HDR Parity Error Indicates when an HDR parity error or framing error on a message from the controller occurs. The corresponding command or data that has the error is usually in the RX buffer, which can be read using Target Read Data Byte (SRDATAB). **NOTE** This field behaves differently for register reads and writes. When reading 0b - No HDR parity error 1b - HDR parity error When writing 0b - No effect 1b - Clear the flag
8 SPAR	SDR Parity Error Indicates when an SDR parity error on a message from the controller occurs. This error also sets the GETSTATUS protocol error field (which becomes 0 after a GETSTATUS read). For read operations, this field becomes 1 when a read abort (timeout) occurs because of the controller not driving clock for more than 100 µs during an SDR read. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No SDR parity error 1b - SDR parity error When writing 0b - No effect 1b - Clear the flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-5 —	Reserved
4 INVSTART	Invalid Start Error Indicates an invalid condition with SCL falling before SDA falls, so there is no start. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No invalid start error 1b - Invalid start error When writing 0b - No effect 1b - Clear the flag
3 TERM	Terminated Error Indicates when the controller terminates a read from a target, if an END is not set by the target (on the same read or the previous read). **NOTE** This field behaves differently for register reads and writes. When reading 0b - No terminated error 1b - Terminated error When writing 0b - No effect 1b - Clear the flag
2 URUNNACK	Underrun and Not Acknowledged (NACKed) Error Indicates when the internal to-bus buffer or FIFO is underrun in the read header and so the module NACKed the header. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No underrun; not acknowledged error 1b - Underrun; not acknowledged error When writing

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Clear the flag
1 URUN	Underrun Error Indicates when the internal to-bus buffer or FIFO is underrun during data read (the application is not providing the data fast enough). The END field or register must be used if the read was the last one. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No underrun error 1b - Underrun error When writing 0b - No effect 1b - Clear the flag
0 ORUN	Overrun Error Indicates when the internal from-bus buffer or FIFO has overrun (arrival of too many characters that you cannot process fast enough). Cutting off the RX_FIFO too close to the boundary when DUT is a target for SDR write or HDR-DDR write can cause an ORUN error when the FIFO is full. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No overrun error 1b - Overrun error When writing 0b - No effect 1b - Clear the flag

72.7.10 Target DMA Control (SDMACTRL)

Offset

Register	Offset
SDMACTRL	20h

Function

Allows DMA to be used for inbound and outbound messages. This register is limited in value for target use because the target must be reactive. These are the two common use case models:

- To avoid an overrun in from-bus collection: if SCONFIG[MATCHSS] becomes 1, then the processor enables the interrupts for START and STOP and enables the DMA to collect the data. The START or STOP interrupt only occurs after a message is directed to the target (MATCHED is 1). The DMA copied data can then be examined.

  NOTE

  Do not enable DMA after a transaction starts. To avoid an RX FIFO overrun, DMA must be enabled only when enabling the target and interrupts.

- To perform larger reads from the to-bus: I3C and I2C reads are preceded by a write that indicates what will be read (or in response to an IBI from the target). Because of this process, the DMA can be used to push through the data:
  - For I3C, the processor must manage the last value, unless the DMA moves wider words and is able to write 1 to the END field. These values are 16-bit values when in Byte mode or 32-bit values when in Halfword mode.
  - For I2C, the controller determines the last value, so the DMA may end early or run out when the controller expects more values.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0		DMAWIDTH		DMATB		DMAFB	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-6 —	Reserved
5-4 DMAWIDTH	Width of DMA Operations Indicates the width of DMA operations, if configured to allow halfword data accesses. 00b,01b - Byte 10b - Halfword (16 bits) (this value ensures that two bytes are available in the FIFO) 11b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 DMATB	DMA Write (To-Bus) Trigger Enables DMA writes. If this field is enabled, it starts a request DMA on a transmit trigger (see Target Data Control (SDATACTRL)). DMATB requests until full, unless the DMA is set up as a trigger. DMATB becomes 0 when MSTATUS[ERRWARN] becomes 1. If you write 1b to this field, DMA is enabled for one frame (ended by DMA or terminated). DMATB automatically becomes 0 after a STOP or START. See SCONFIG[MATCHSS] for more information. If you write 10b to this field, DMA is enabled until turned off. The value must only be used with Controller Message mode. 00b - DMA not used 01b - DMA enabled for one frame 10b - DMA enabled until turned off 11b - Reserved
1-0 DMAFB	DMA Read (From-Bus) Trigger Enables DMA reads. If this field is enabled, DMAFB requests DMA on receive trigger (see SDATACTRL). It requests until empty, unless the DMA is set up as a trigger. This field becomes 0 when SSTATUS[ERRWARN] becomes 1. If this field is 1b (DMA is enabled for one frame), it automatically becomes 0 on STOP or Repeated START. See SCONFIG[MATCHSS] for more information. **NOTE** Do not enable DMA after a transaction starts. It must be enabled only when enabling the target and interrupts to avoid any kind of RX FIFO overrun. 00b - DMA not used 01b - DMA enabled for one frame 10b - DMA enabled until turned off 11b - Reserved

72.7.11 Target Data Control (SDATACTRL)

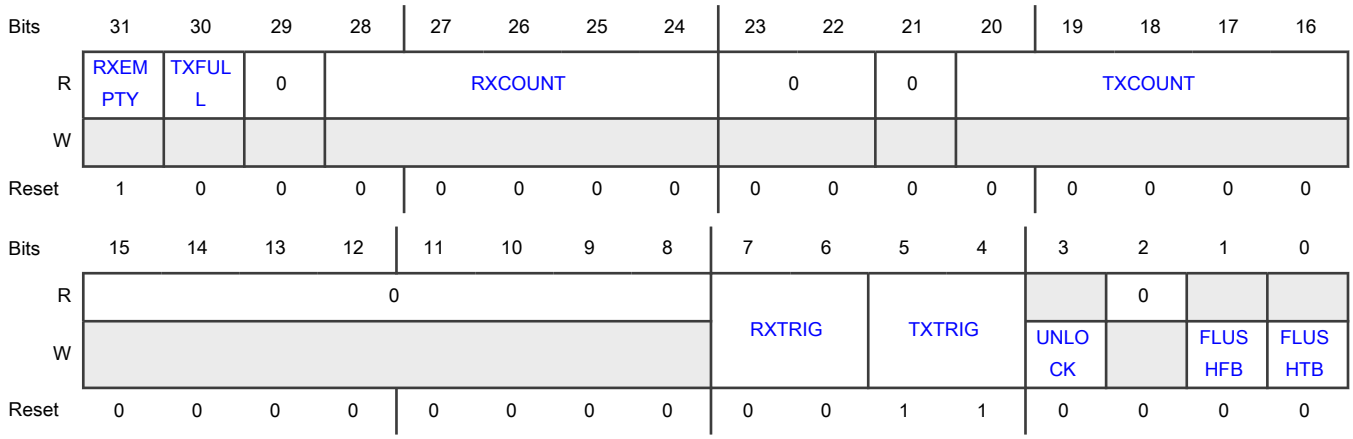
Offset

Register	Offset
SDATACTRL	2Ch

Function

Assists in data control when no FIFO is used. Also assists in data control of FIFO when the FIFO is available (regardless of size) allowing some control over the FIFO behavior. This register controls when to interrupt based on fullness or emptiness of a buffer or FIFO. It also controls behavior related to width, when the buffer or FIFO is not 1 byte wide.

Diagram



Fields

Field	Function
31 RXEMPTY	Receive is Empty 0b - Not empty 1b - Empty
30 TXFULL	Transmit is Full 0b - Not full 1b - Full
29 —	Reserved
28-24 RXCOUNT	Count of Bytes in Receive
23-22 —	Reserved
21 —	Reserved
20-16 TXCOUNT	Count of Bytes in Transmit
15-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-6 RXTRIG	Receive Trigger Level Indicates the trigger level for receive fullness when using a FIFO. The field affects the RXPEND interrupt. 00b - Trigger when not empty 01b - Trigger when 1/4 or more full 10b - Trigger when 1/2 or more full 11b - Trigger when 3/4 or more full
5-4 TXTRIG	Transmit Trigger Level Indicates the trigger level for transmit emptiness when using a FIFO. The field affects the TXNOTFULL interrupt. 00b - Trigger when empty 01b - Trigger when 1/4 full or less 10b - Trigger when 1/2 full or less 11b - Default (trigger when 1 less than full or less)
3 UNLOCK	Unlock Indicates whether the RXTRIG and TXTRIG fields can be changed on a write. This field must be 1 in the same cycle while writing to TXTRIG or RXTRIG. 0b - Cannot be changed 1b - Can be changed
2 —	Reserved
1 FLUSHFB	Flush From-Bus Buffer or FIFO Not normally used.
0 FLUSHTB	Flush To-Bus Buffer or FIFO Indicates when the controller terminates a to-bus (read) message prematurely.

72.7.12 Target Write Data Byte (SWDATAB)

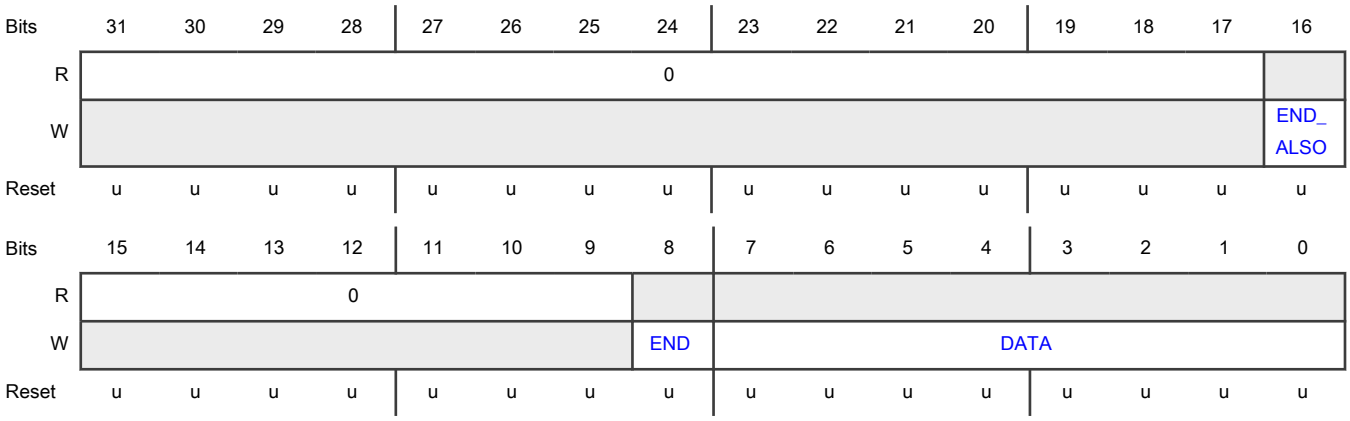
Offset

Register	Offset
SWDATAB	30h

Function

Allows writing a byte to the bus (to the controller) unless an external FIFO is used. Writing a byte requires a byte plus an end-of-data (last) marker bit. A byte must not be written unless there is room, indicated by SSTATUS[TXNOTFULL] = 1.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 END_ALSO	End Also Marks the last byte of the message. This field is required for I3C, but is optional for I2C. If this field is 0, it indicates that there are more bytes in the message. For HDR-DDR, the byte with END_ALSO must be an even byte (second, fourth, sixth, and so on) because DDR uses byte pairs. 0b - Not the end 1b - End
15-9 —	Reserved
8 END	End Marks the last byte of the message. If this field is 0, it indicates that there are more bytes in the message. This field is required for I3C, but is optional for I2C. For HDR-DDR, the byte with END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte pairs. 0b - Not the end 1b - End
7-0 DATA	Data Indicates the data byte to be sent to the controller.

72.7.13 Target Write Data Byte End (SWDATABE)

Offset

Register	Offset
SWDATABE	34h

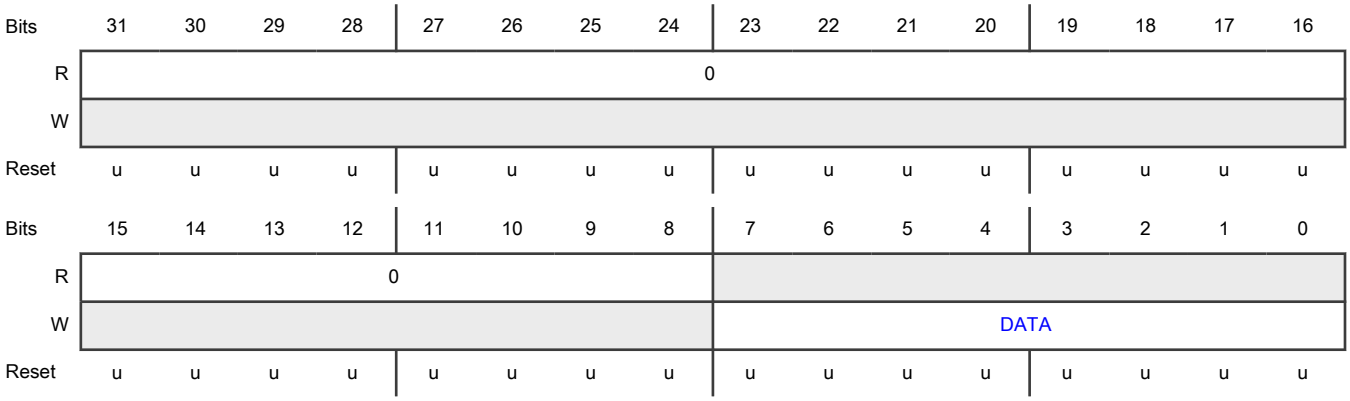
Function

Allows writing a byte to the bus (to the controller) unless an external FIFO is used.

Unlike Target Write Data Byte (SWDATAB), writing a byte only requires the byte itself, and is marked as end-of-data (last byte). A byte must not be written unless there is room, indicated by SSTATUS[TXNOTFULL] = 1.

For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte pairs.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 DATA	Data Indicates the data byte to be sent to the controller.

72.7.14 Target Write Data Halfword (SWDATAH)

Offset

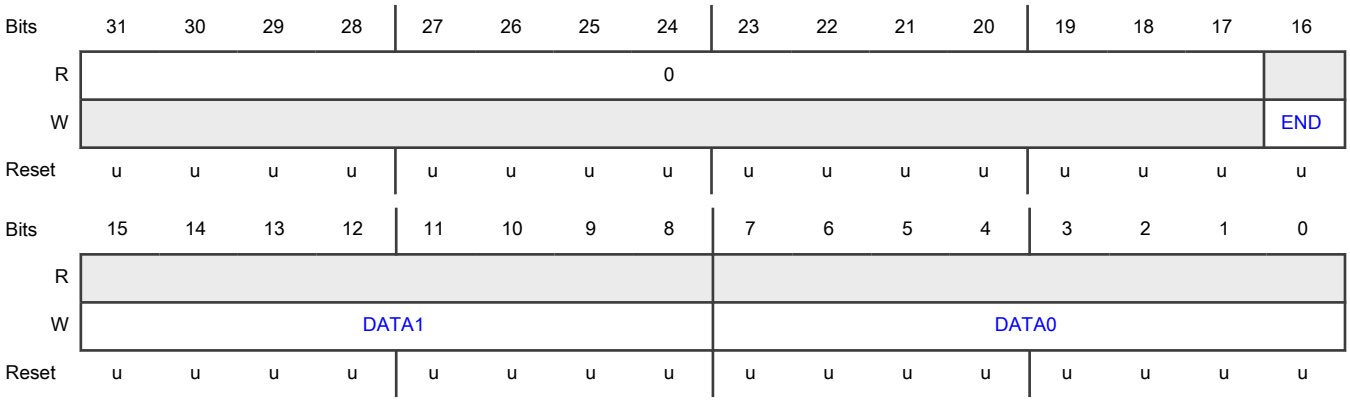
Register	Offset
SWDATAH	38h

Function

Allows writing a halfword (pair of bytes) to the bus unless an external FIFO is used. The low byte is followed by the high byte. The 16th bit marks the end; that is, the last byte of the halfword is the end.

An end-of-data (last) marker bit is allowed (or must be 0). A halfword must not be written unless there is room for both, as indicated by the use of transmit FIFO level trigger or SDATACTRL[TXCOUNT].

Diagram



Fields

Field	Function
31-17 —	Reserved
16 END	End of Message Marks the last byte of the message. This field always marks DATA1 as the end. The field is required for I3C, but is optional for I2C. If this field is 0, it indicates that there are more bytes in the message. For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte pairs. 0b - Not the end 1b - End
15-8 DATA1	Data 1 Indicates the second byte to be sent to the controller.
7-0 DATA0	Data 0 Indicates the first byte to be sent to the controller.

72.7.15 Target Write Data Halfword End (SWDATAHE)

Offset

Register	Offset
SWDATAHE	3Ch

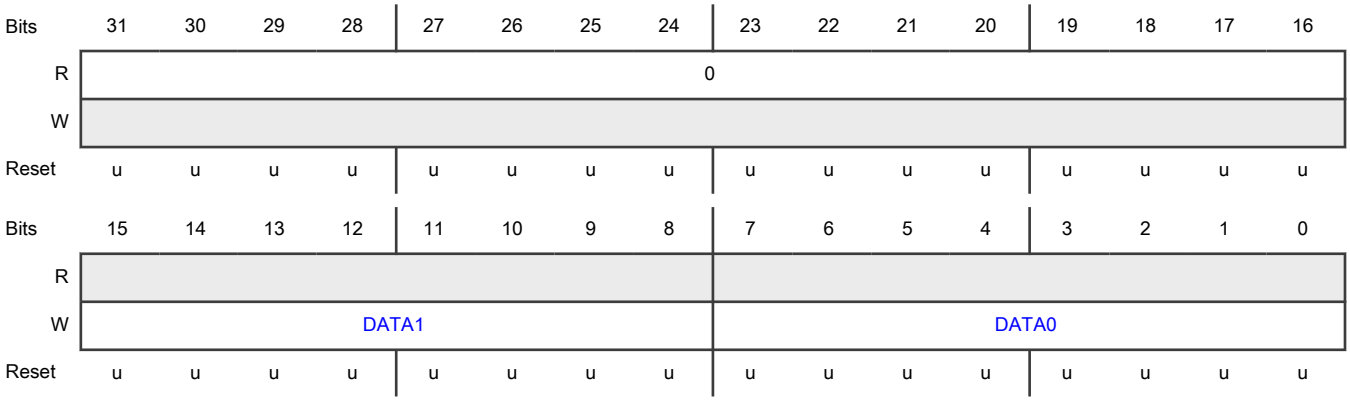
Function

Allows writing a halfword of data, which is the end (the last byte of the halfword is the end). The target writes the halfword (byte pair) in the same way it writes to Target Write Data Halfword (SWDATAH), but marks the second byte as end-of-data (last byte).

For HDR-DDR, the byte with the END must be even (second, fourth, sixth, and so on) because DDR uses byte pairs.

A halfword must not be written unless there is room for both, as indicated by the use of transmit FIFO level trigger or SDATACTRL[TXCOUNT].

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 DATA1	Data 1 Indicates the second byte to be sent to the controller.
7-0 DATA0	Data 0 Indicates the first byte to be sent to the controller.

72.7.16 Target Read Data Byte (SRDATAB)

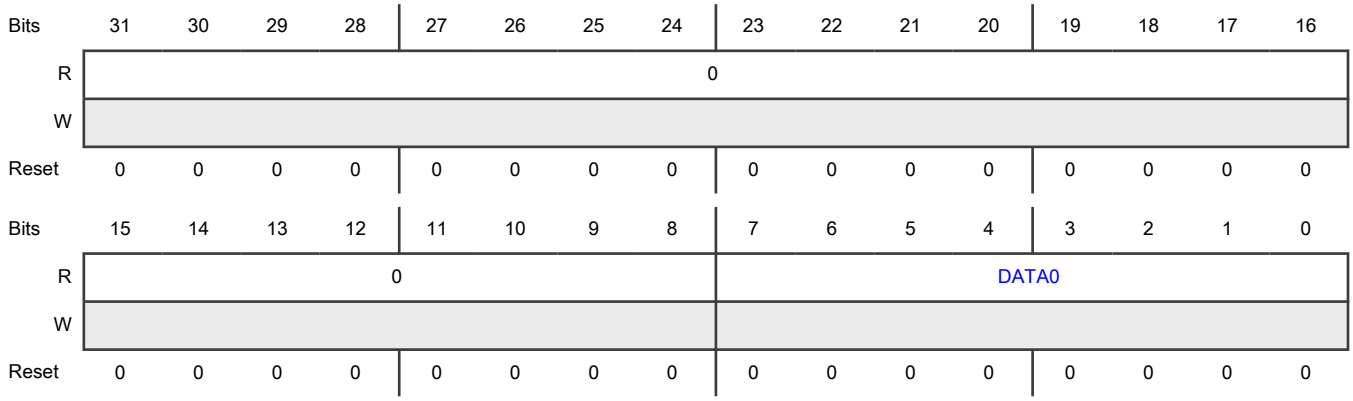
Offset

Register	Offset
SRDATAB	40h

Function

Allows reading a byte from the bus (controller). A byte must not be read unless there is data waiting, as indicated by SSTATUS[RX_PEND] = 1.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 DATA0	Data 0 Indicates the byte read from the controller.

72.7.17 Target Read Data Halfword (SRDATAH)

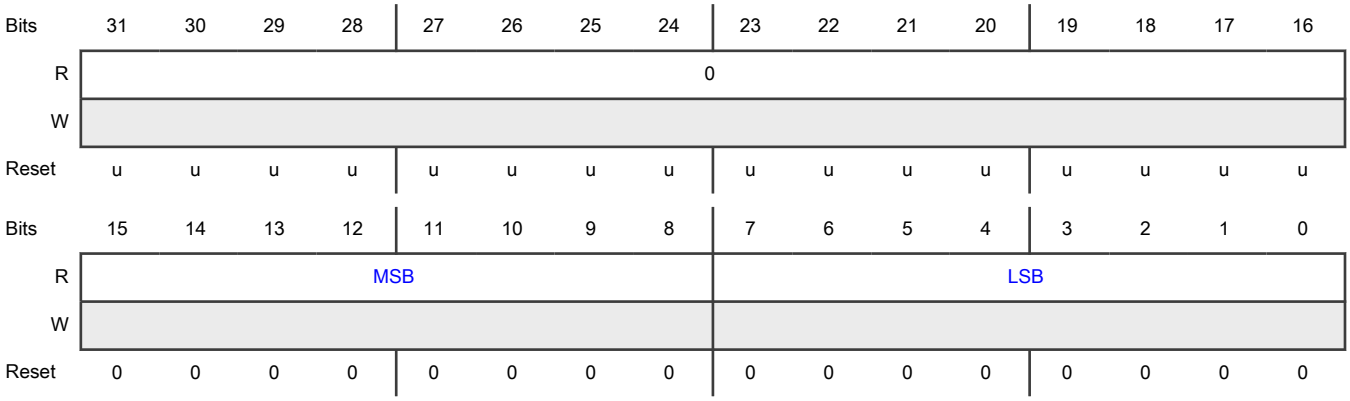
Offset

Register	Offset
SRDATAH	48h

Function

Allows reading a halfword (byte pair) written by the target after an SDR read or DAA or DDR. This register is used only when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively. A halfword must not be read unless there are at least two bytes of data waiting, as indicated by the use of receive FIFO level trigger or SDATACTRL[RXCOUNT].

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MSB	High Byte Indicates the second byte read from the target.
7-0 LSB	Low Byte Indicates the first byte read from the target.

72.7.18 Target Write Data Byte (SWDATAB1)

Offset

Register	Offset
SWDATAB1	54h

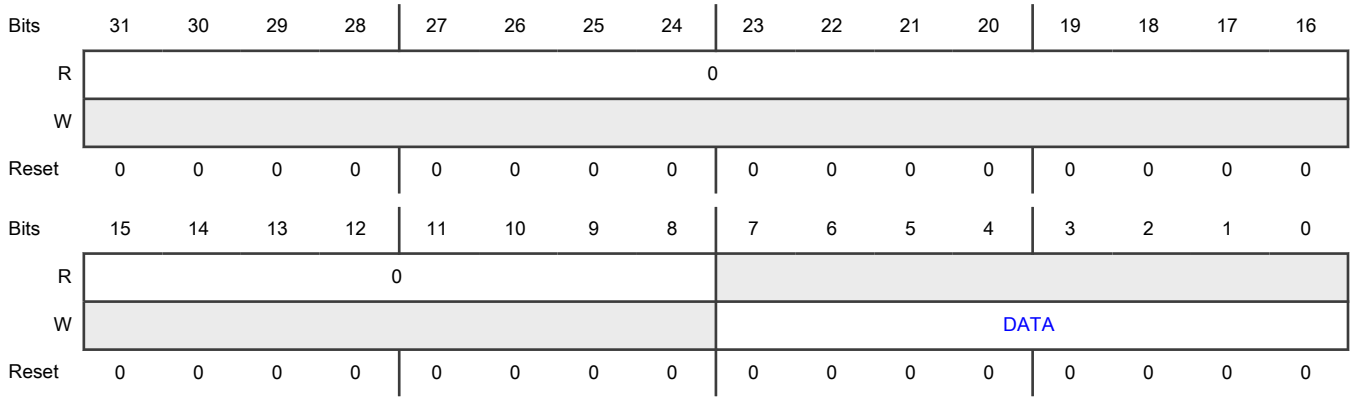
Function

Allows writing a single byte to the bus (to the controller) in a way that only bits 7:0 are used.

This register is intended to be used by DMAs (the upper bytes of the APB word are ignored).

A byte must not be written unless there is room, as indicated by SSTATUS[TXNOTFULL] = 1.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 DATA	Data Indicates the byte to be sent to the controller.

72.7.19 Target Capabilities 2 (SCAPABILITIES2)

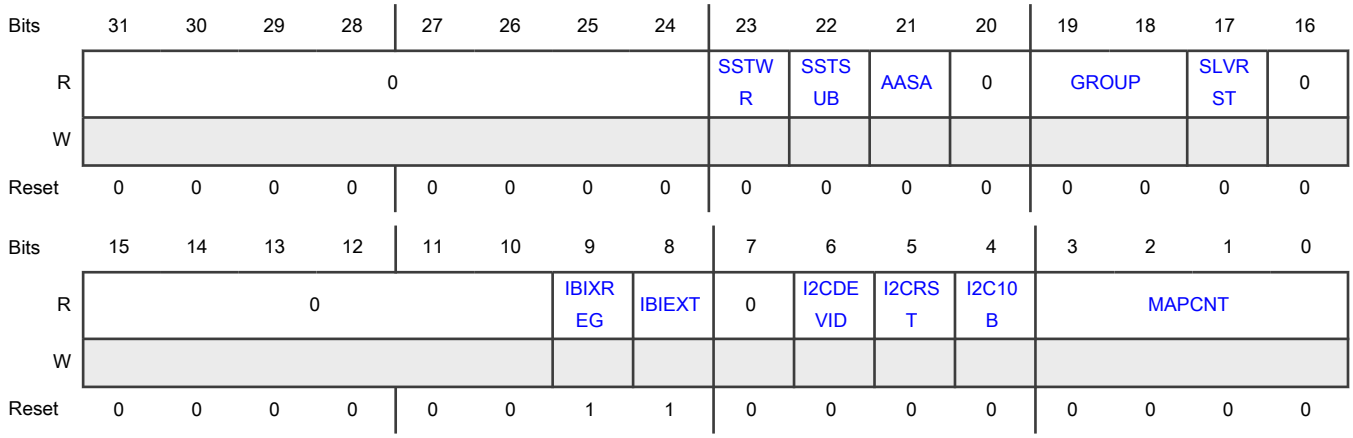
Offset

Register	Offset
SCAPABILITIES2	5Ch

Function

Indicates which features are available and supported in this module, including controller and target capabilities, HDR modes, and others.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 SSTWR	Target-Target(s)-Tunnel Write Capable Indicates whether target-target(s)-tunnel is write capable. 0b - Not write capable 1b - Write capable
22 SSTSUB	Target-Target(s)-Tunnel Subscriber Capable Indicates whether target-target(s)-tunnel is subscriber capable. 0b - Not subscriber capable 1b - Subscriber capable
21 AASA	SETAASA Supports the set static address as dynamic address CCC (SETAASA) feature. 0b - SETAASA not supported 1b - SETAASA supported
20 —	Reserved
19-18 GROUP	Group Indicates whether v1.1 group addressing is supported. Groups use mapping, so MAPCNT must be the same or greater than GROUP. 00b - v1.1 group addressing not supported 01b - One group supported

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Two groups supported 11b - Three groups supported
17 SLVRST	Target Reset Indicates whether v1.1 target reset is supported. 0b - Not supported 1b - Supported
16 —	Reserved
15-10 —	Reserved
9 IBIXREG	In-Band Interrupt Extended Register Supports extended registers for IBIs. This field indicates whether Extended IBI Data 1 (IBIEXT1) is available. Extended IBI Data 2 (IBIEXT2) is available if IBIEXT1[MAX] > 3. 0b - Not supported 1b - Supported
8 IBIEXT	In-Band Interrupt EXTDATA Supports SCTRL[EXTDATA] to allow data beyond the mandatory data byte (MDB) for IBIs. 0b - Not supported 1b - Supported
7 —	Reserved
6 I2CDEVID	I2C Device ID Indicates whether I2C device ID is supported. 0b - Not supported 1b - Supported
5 I2CRST	I2C Software Reset Supports I2C software reset. 0b - Not supported 1b - Supported
4	I2C 10-bit Address

Table continues on the next page...

Table continued from the previous page...

Field	Function
I2C10B	Supports 10-bit I2C address in MAP 1. If this field is 1, MAPCNT must be 1 or greater. 0b - Not supported 1b - Supported
3-0 MAPCNT	Map Count Indicates the number of maps that are allowed. If there is no mapping, this field is 0.

72.7.20 Target Capabilities (SCAPABILITIES)

Offset

Register	Offset
SCAPABILITIES	60h

Function

Indicates which features are available and supported in this module, including controller and/or target capabilities, HDR modes, and others.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMA	INT	FIFORX		FIFOTX		EXTFIFO		0		TIMECTRL		IBI_MR_HJ			
W																
Reset	1	1	1	0	1	0	0	0	0	0	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CCCHANDLE				SADDR		MASTER	0	HDRSUPP		IDREG				IDENA	
W																
Reset	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0

Fields

Field	Function
31 DMA	Direct Memory Access Indicates whether DMA is supported. 0b - Not supported 1b - Supported

Table continues on the next page...

Table continued from the previous page...

Field	Function
30 INT	Interrupts Indicates whether interrupts are supported. 0b - Not supported 1b - Supported
29-28 FIFORX	FIFO Receive Indicates the size of receive (from-bus) FIFO. **NOTE** Controller and target use the same receive FIFO, so size is the same for the controller receive FIFO as well. FIFO size of SDR and HDR-DDR is in bytes. 00b - Two or three 01b - Four 10b - Eight 11b - 16 or larger
27-26 FIFOTX	FIFO Transmit Indicates the size of transmit (to-bus) FIFO. **NOTE** Controller and target use the same transmit FIFO, so size is the same for the controller transmit FIFO as well. FIFO size of SDR and HDR-DDR is in bytes. 00b - Two 01b - Four 10b - Eight 11b - 16 or larger
25-23 EXTFIFO	External FIFO Indicates whether external FIFOs are enabled. If they are not enabled, then check FIFOTX and FIFORX for the internal FIFO. 000b - No external FIFO available 001b - Standard available or free external FIFO 010b - Request track external FIFO All other values are reserved.
22 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 TIMECTRL	Time Control Specifies whether any time-control type is supported. 0b - No time control supported 1b - At least one time-control type supported
20-16 IBI_MR_HJ	In-Band Interrupts, Controller Requests, Hot-Join Events Indicates which events (IBI, CR, and HJ) are allowed. For example, if this field is 00011b, IBI (bit 0) and IBI_HAS_DATA (bit 1) functionality are both enabled. 0_0000b - Application cannot generate IBI, CR, or HJ 1_xxxxb - Application can use SCONFIG[BAMATCH] for bus-available timing x_1xxxb - Application can generate a Hot-Join event x_x1xxb - Application can generate a controller request for a secondary controller x_xx1xb - When bit 0 = 1, the IBI has data from the SCTRL register x_xxx1b - Application can generate an IBI
15-12 CCCHANDLE	Common Command Codes Handling Indicates what manages CCC between I3C and the application. 0000b - All handling features disabled 1xxxb - GETSTATUS CCC returns the value of SCTRL[VENDINFO] x1xxb - GETSTATUS CCC returns the values of SCTRL[PENDINT] and SCTRL[ACTSTATE] xx1xb - The I3C module manages maximum read and write lengths, and max data speed xxx1b - The I3C module manages events, activities, status, HDR, and if enabled for it, ID and static-address-related items
11-10 SADDR	Static Address Indicates how the static address is managed. 00b - No static address 01b - Static address is fixed in hardware 10b - Hardware controls the static address dynamically (for example, from the pin strap) 11b - SCONFIG register supplies the static address
9 MASTER	Controller Specifies whether controller capability is supported. 0b - Not supported 1b - Supported
8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-6 HDRSUPP	High Data Rate Support Indicates which HDR modes are supported. 00b - No HDR modes supported 01b - DDR mode supported All other values are reserved.
5-2 IDREG	ID Register Indicates which ID features are available in the configurable registers. 0000b - All ID register features disabled 1xxxb - A Bus Characteristics Register (BCR) is available x1xxb - A Device Characteristic Register (DCR) is available xx1xb - An ID Random field is available xxx1b - ID Instance is a register; used if there is no PARTNO register
1-0 IDENA	ID 48b Handler Indicates what handles the 48-bit ID value. 00b - Application 01b - Hardware 10b - Hardware, but the I3C module instance handles ID 48b 11b - A part number register (PARTNO)

72.7.21 Target Dynamic Address (SDYNADDR)

Offset

Register	Offset
SDYNADDR	64h

Function

Fills with the assigned address after the controller has assigned it via SETDASA or ENTDAA CCC commands. It clears if the RESETDAA CCC is used. The current validity state is also indicated through SSTATUS[DAVALID], which can be used to interrupt the processor. The DCAUSE field can be used to determine the changes, if enabled.

NOTE

If mapped addresses are enabled, they are accessed through the SMAPCTRL*n* registers.

The first dynamic address (DA[0]) may be written irrespective of whether the mapping is enabled, if configured to allow writes used to restore the DA after a power-down (an ultra-low power state that loses power to peripherals but retains the DA somewhere else). This is not required if state-retention flops are used for the DA. This mechanism only allows writes when target is disabled

(and is ignored otherwise). If the controller uses RSTDAA or SETNEWDA, then it overrides this mechanism and yields to the controller-assigned DA. When enabling the target, SCONFIG[OFFLINE] must be 1. This waits for evidence that the bus is not in I3C HDR mode and exits when an HDR exit pattern is seen or when 60 µs has expired. This makes it safe to monitor START and STOP. If the application needs to do an IBI, then the application must either wait for a STOP (see STATUS) or make sure that 200 µs have gone by with no activity (no START or STOP) before the application emits the IBI.

The MAPIDX or MAPSA model allows writing additional DAs (and SAs) into a list (the number in the list is preconfigured); any DA or SA with DAVALID = 0 are never matched. The additional DAs may be based on the bridge target CCC or done by a move (read current DA and then copy into the upper map, and then invalidate the 0 map) when matching ENTDAA over and over. Any mapped location can be invalidated by writing MAPIDX and DAVALID = 0. The mapped ones are not reset by the RSTDA CCC (see Target Message Map Address (SMSGMAPADDR)).

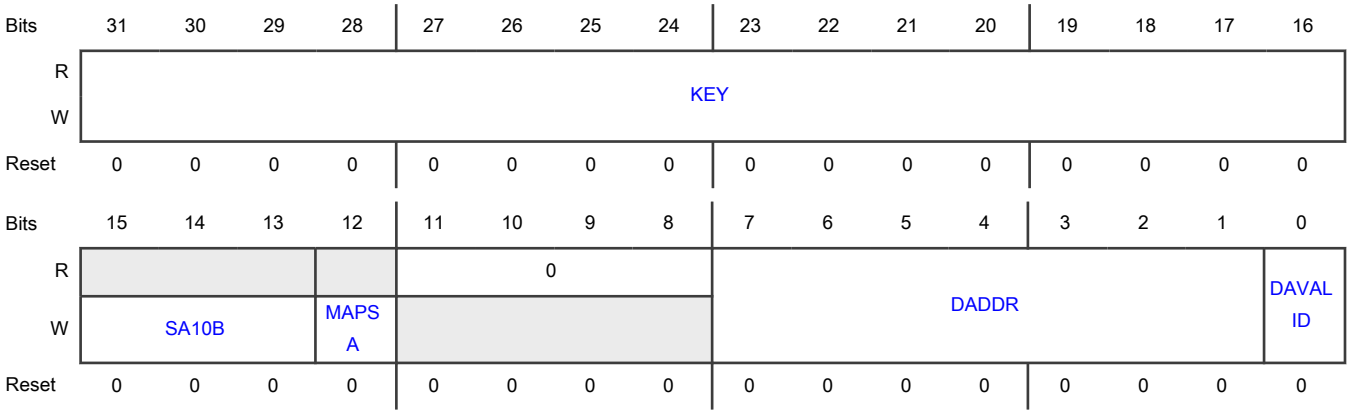
To copy the base DA to the mapped set, you must perform the configuration to allow it; otherwise it leads to distinct mechanisms:

- If used in I3C mode, a base DA is required, so the last one must not be cleared (or writing a dynamic address with DAVALID = 1 is necessary).

NACK/ACK of mapped DAs/SAs: The register also permits specific DAs (above the base) to NACK or ACK writes or reads, such as for bridges when the endpoint is busy. The NACK must not be set for too long, or the controller may consider the target in an error state. The model is to write with KEY = A731h and the MAPIDX and DAVALID to select ACK or NACK.

Mapping: If MAPIDX ≠ 0 and KEY = 0, then the next read returns details on the static or dynamic address at that location. If the read does not have MAPIDX ≠ 0, then it is not an acceptable location.

Diagram



Fields

Field	Function
31-16 KEY	Key Specifies the value of KEY. Must be set to A4D9h to write to the DADDR field (and 1 to the DAVALID field). Write only when a target is not enabled (for restoring after power-down sleep with auto-restore). The mapped locations and base may be written when a target is enabled, but do not write when there are transactions on the I3C bus. KEY reads back 1 if overwritten, else KEY is 0 if assigned by the controller including when the controller changes it. If address mapping is allowed, then writing with KEY = CB19h is used to clear the base DA.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If mapping is allowed, then writing with the key of A731h means that MAPIDX indicates which mapped address to set to ACK or NACK, such that DAVALID = 1 if ACK, DAVALID = 0 if NACK. This allows having the address refuse write and read requests when NACK. When using read back from map index (write with MAPIDX ≠ 0, KEY = 0, read) bit [16] = 1 if NACK, else 0.
15-13 SA10B	10-Bit Static Address Stores a 10-bit static address composed of {SA10B,DADDR}, if not 0 when MAPSA is set. Only one static address may be stored this way, and it must be MAPIDX == 1.
12 MAPSA	Map a Static Address Sets a static address into the list if MAPSA = 1 on a write with MAPIDX ≠ 0; otherwise, a dynamic address is used.
11-8 —	Reserved
7-1 DADDR	Dynamic Address Specifies the dynamic address (DA). This is the assigned dynamic address when DAVALID is 1 and is a static address if the MAPSA field is 1.
0 DAVALID	Dynamic Address Valid Determines whether a dynamic address is assigned. This field is 1 for ACK and 0 for NACK when using the A731h KEY. 0b - DANOTASSIGNED: a dynamic address is not assigned 1b - DAASSIGNED: a dynamic address is assigned

72.7.22 Target Maximum Limits (SMAXLIMITS)

Offset

Register	Offset
SMAXLIMITS	68h

Function

Indicates the limits set by the controller (or the originally requested limits). The maximum limits are not enabled in the hardware design, including maximum read and write lengths. If the maximum read and write lengths are enabled, then the current setting (including the default request) shows up in this register (SMAXLIMITS).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				MAXWR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				MAXRD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-16 MAXWR	Maximum Write Length Indicates the maximum write length, which must be between 8 to 4095 (saturation). The application must not set the maximum write length to a higher value than the maximum write length set by the controller.
15-12 —	Reserved
11-0 MAXRD	Maximum Read Length Indicates the maximum read length, which must be between 16 to 4095 (saturation). The application must not set the maximum read length to a higher value than the maximum read length set by the controller.

72.7.23 Target ID Part Number (SIDPARTNO)

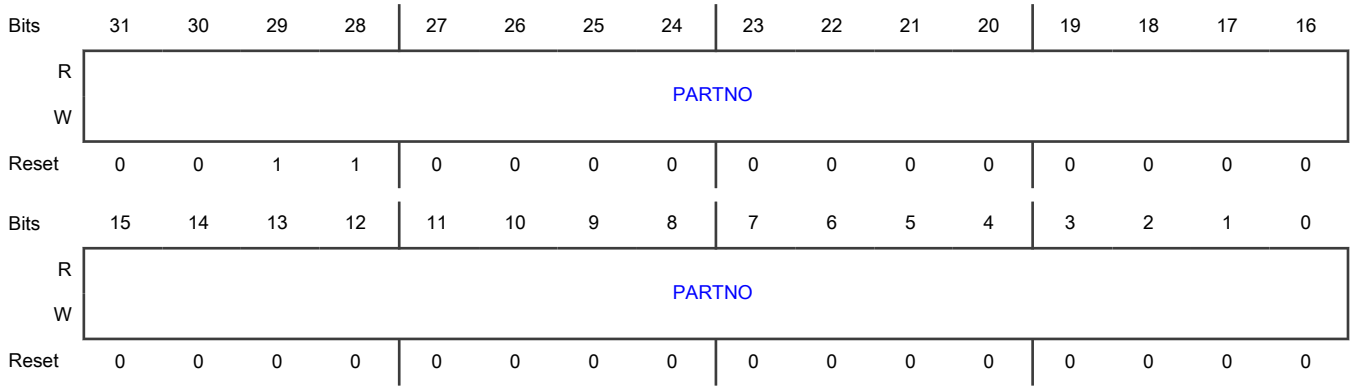
Offset

Register	Offset
SIDPARTNO	6Ch

Function

Allows you to write the ID part number. You must write a nonzero value into the PARTNO field because 0 is not valid.

Diagram



Fields

Field	Function
31-0 PARTNO	Part Number

72.7.24 Target ID Extension (SIDEXT)

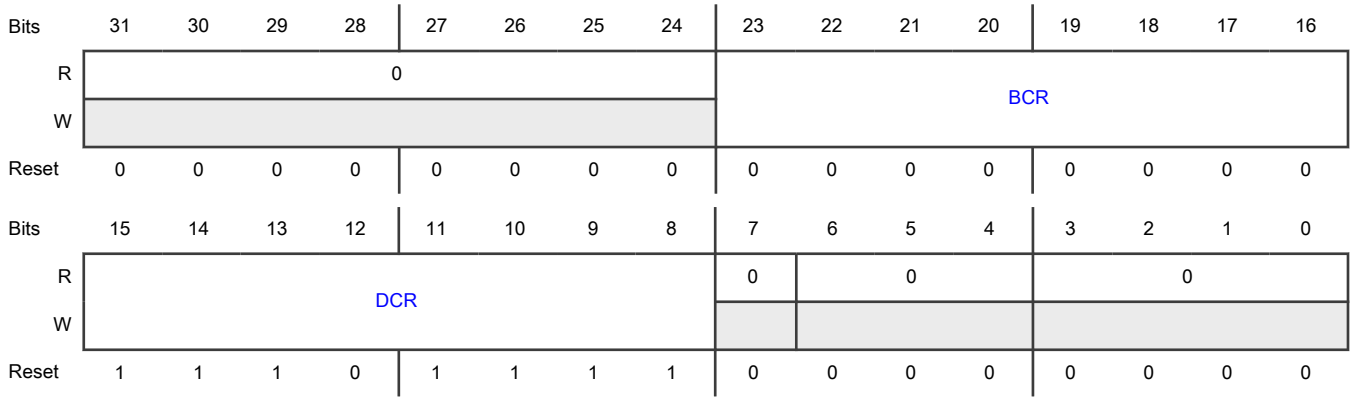
Offset

Register	Offset
SIDEXT	70h

Function

Allows you to write the ID extension of DCR and/or BCR.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 BCR	Bus Characteristics Register Sets the value for BCR, if this field is configured. Otherwise, the default value is considered. This field controls features such as secondary controller and slow-speed requirements.
15-8 DCR	Device Characteristic Register Sets the value for DCR, if this field is configured. Otherwise, the default value is considered.
7 —	Reserved
6-4 —	Reserved
3-0 —	Reserved

72.7.25 Target Vendor ID (SVENDORID)

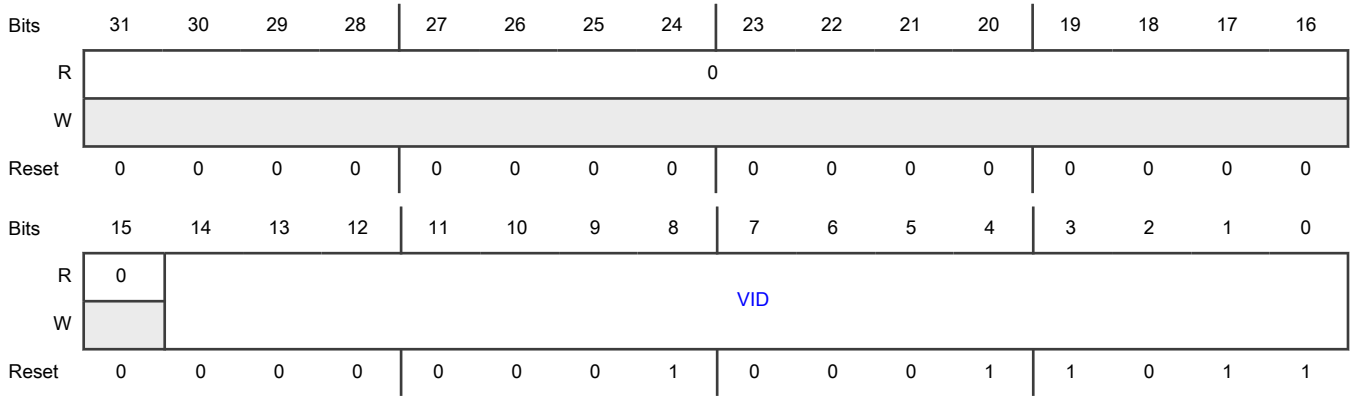
Offset

Register	Offset
SVENDORID	74h

Function

Allows you to write the vendor ID. The default value is the chip vendor ID, and is set from the constant field. When using the chip vendor ID, the part number (PARTNO) does not collide with other uses. The MIPI vendor ID is available to all companies (MIPI membership is not required). To get a vendor ID, make a request at the mipi.org website.

Diagram



Fields

Field	Function
31-15 —	Reserved
14-0 VID	Vendor ID Configures the 15-bit MIPI vendor ID. If not configured, the default value is considered.

72.7.26 Target Time Control Clock (STCCLOCK)

Offset

Register	Offset
STCCLOCK	78h

Function

Allows you to dynamically set the time control clock and accuracy information. The clock frequency and accuracy are constants set by the hardware. If the clock can be adjusted (that is, divided) or if the accuracy could vary with knowable information, then the clock may be set via this register, which must be updated whenever the clock source is changed.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FREQ								ACCURACY							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0

Fields

Field	Function
31-16 —	Reserved
15-8 FREQ	Clock Frequency Indicates the clock frequency in 0.5 MHz steps. For example, a value of 20 in this field indicates a frequency of 10 MHz. Default set by parameters if configured.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 ACCURACY	Clock Accuracy Indicates the clock accuracy in 1/10ths of %. For example, a value of 15 indicates an accuracy of 1.5%. Default set by parameters if configured.

72.7.27 Target Message Map Address (SMSGMAPADDR)

Offset

Register	Offset
SMSGMAPADDR	7Ch

Function

Allows you to determine which address is matched when STATUS[MATCHED] = 1. This register is used when the DYNADDR register builds a list of extra DAs or SAs to match. It holds the last three matches.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												MAPLASTM2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				MAPLASTM1				0			LASTSTA...	MAPLAST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-16 MAPLASTM2	Matched Previous Index 2 Indicates index 2 of the previous matched address (0 for the base address).
15-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 MAPLASTM1	Matched Previous Address Index 1 Indicates index 1 of the previous matched address (0 for the base address).
7-5 —	Reserved
4 LASTSTATIC	Last Static Address Matched Indicates whether the last matched address was an I2C static address or an I3C dynamic address. 0b - I3C dynamic address 1b - I2C static address
3-0 MAPLAST	Matched Address Index Indicates the matched address index for current or last matched message (0 for the base address).

72.7.28 Controller Control (MCTRL)

Offset

Register	Offset
MCTRL	84h

Function

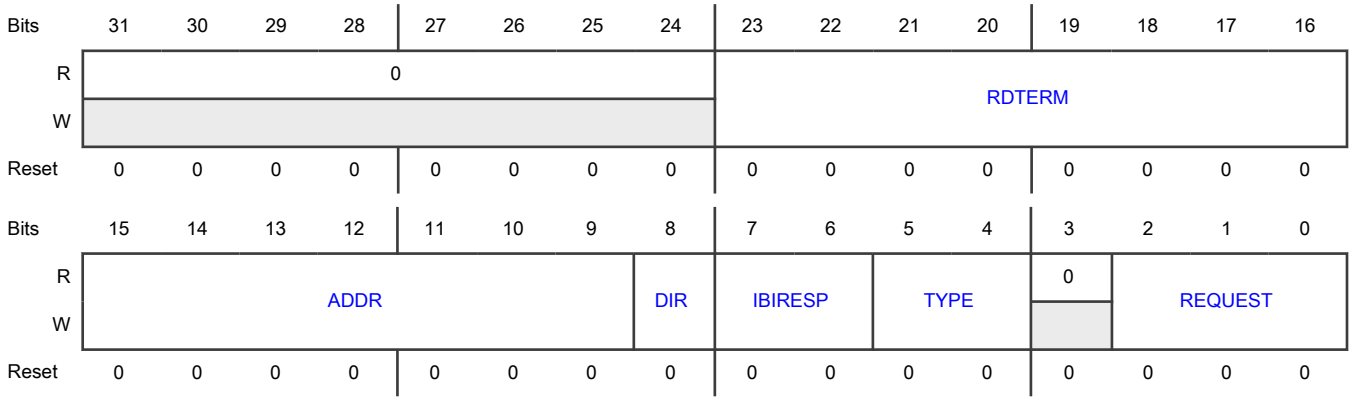
Starts activities on the I3C or I2C bus (see the MWMSG registers). A request cannot be changed when a message is in progress; the REQUEST field becomes 0 automatically.

You must write to MCTRL fields as per use case or as mentioned in Operating modes. Bit read and write checking may not work independently.

NOTE

If MCONFIG[MSTENA] is configured for I2C Controller mode (legacy I2C), only REQUEST = 1 and REQUEST = 2 are accepted. Also, fields are constrained to I2C-supported fields.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 RDTERM	Read Terminate Counter Determines when to terminate a read operation: • For I2C, this field controls when to NACK a read. • For I3C, this field can be used to terminate (end) a read: — If RDTERM is 0, it has no effect. — If RDTERM is 1, the read terminates after the next character. — If RDTERM is 2, the read terminates after the next two characters. The field supports up to 255 characters. In DDR mode, RDTERM terminates the read based on word counts (for DDR) instead of byte counts (for SDR). The field self-clears when MSTATUS[COMPLETE] becomes 1. **NOTE** You may write 1 to this field anytime, but a value greater than 1 must be written when starting EmitStartAddress.
15-9 ADDR	Address Indicates the type of address: I3C dynamic address or I2C static address. Some values are not allowed based on the bus that is used (I3C or I2C).
8 DIR	Direction Indicates the direction, write or read. 0b - Write 1b - Read
7-6	In-Band Interrupt Response

Table continues on the next page...

Table continued from the previous page...

Field	Function
IBIRESP	Indicates the response to use when you get an IBI from START, and when to force using an IBI ACK NACK request when completing a manual IBI. Completion of a manual IBI means that the target DA is known, and so the mandatory byte (or not) is specified by the application when acknowledging. This field is also used when a message is emitted in Message mode using the MWMSG_SDR or MWMSG_DDR registers. If an IBI with MDB (mandatory byte) is ACKed, the controller limits it to a maximum of eight more bytes after the MDB. RDTERM = 1 can be used with REQUEST = NONE to terminate data from a target sooner. If this field is 0b, it indicates ACK (acknowledge). When REQUEST = 1 (EmitStartAddr) or REQUEST = 7 (AutoIBI), Controller In-band Interrupt Registry and Rules (MIBIRULES) decides whether to ACK with mandatory byte. When REQUEST = 3 (IBIAckNack), ACK with no mandatory byte. If this field is 1b, it indicates NACK (reject) or no acknowledgement. If this field is 10b, it indicates acknowledge with mandatory byte. When REQUEST = 1 or REQUEST = 7, ignore Controller In-band Interrupt Registry and Rules (MIBIRULES). Do not use this setting unless only targets with a mandatory byte can cause an IBI. When REQUEST = 3, ACK with mandatory byte. If this field is 11b, it indicates manual. When REQUEST = 1 or REQUEST = 7, stop and wait for a decision using the IBI ACK NACK request. When REQUEST = 3, this field is reserved. **NOTE** A CR and HJ always cause IBIRESP to become 3 (manual) so the application must decide. 00b - ACK (acknowledge) 01b - NACK (reject) 10b - Acknowledge with mandatory byte 11b - Manual
5-4 TYPE	Bus Type with EmitStartAddr Works with REQUEST to determine the bus type. See Determining bus types for more information. The following values are valid only if REQUEST is 1. 00b - I3C 01b - I2C 10b - DDR 11b - Reserved
3 —	Reserved
2-0 REQUEST	Request Emits the requested operation when performing in pieces, instead of performing by message. You must check Controller Status (MSTATUS) because some requests can only be made in some states. For example, the system cannot enter SDR mode from DDR mode, and it cannot use an incorrect request in DAA mode.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If this field is 0b, it indicates that no request is present (NONE). The REQUEST field returns to NONE after finishing a request. Controller Status (MSTATUS) indicates the state of the controller. See AutoIBI mode. NONE is written as 0 only in these cases: when writing 1 to MCTRL[RDTERM] (to stop a read in progress) or when configuring MCTRL[IBIRESP] for MSG use. If this field is 1b, it emits START with address and direction (EMITSTARTADDR), either from Stopped state or in the middle of a single data rate (SDR) message. If from Stopped state (Idle), then EmitStart may be prevented by an event (such as IBI, CR, HJ). In this case, an appropriate interrupt is signaled. EmitStart can be resubmitted. If this field is 10b, it emits a STOP on bus (EMITSTOP). Must be in SDR mode. In DAA mode, emitting STOP exits DAA mode. If this field is 11b, it indicates manual IBI ACK or NACK (IBIACKNACK). When MCTRL[IBIRESP] indicates a hold on an IBI to allow a manual decision, this request completes it. The field uses MCTRL[IBIRESP] to provide the information. If this field is 100b, and if currently not in DAA mode, it emits START, 7E, ENTDAA sequence, and then emits 7E/R to process the first target (PROCESSDAA). Stops just before the new dynamic address (DA) is to be emitted. The DA is written using Controller Write Data Byte (MWDATAB), then process DAA is requested again to write the new address, and then it starts the next unless marked to STOP. An MSTATUS indicating NACK means DA was not accepted (for example, parity error). If PROCESSDAA is NACKed on the 7E/R request, meaning no more targets need a DA, then a COMPLETE is signaled (along with DONE) and a STOP issued. If this field is 110b, it emits an exit pattern from any state (force exit and target reset). End DDR mode (including MSGDDR), including a STOP afterward. If this field is 111b, the target is held in a Stopped state, but autoemits a START, 7E sequence when the target holds SDA low for an IBI (AUTOIBI). Actual IBI handling is defined by MCTRL[IBIRESP]. 000b - NONE 001b - EMITSTARTADDR 010b - EMITSTOP 011b - IBIACKNACK 100b - PROCESSDAA 101b - Reserved 110b - Force Exit and Target Reset 111b - AUTOIBI

72.7.29 Controller Status (MSTATUS)

Offset

Register	Offset
MSTATUS	88h

Function

Indicates the controller status, including which events cause interrupts. The peripherals share the IRQ (called parallel-to-target status).

Because a peripheral can either be in Controller or Target mode, but not both at the same time, only one (target or controller peripheral) can cause the IRQ. If there is an IRQ and the peripheral is a controller, then this register (MSTATUS) indicates the status.

If there is an IRQ and the peripheral is a target, then Target Status (SSTATUS) indicates the status.

This register self-clears if the COMPLETE field becomes 1.

If the MCONFIG register is configured only for I2C, some states and fields never remain active (for example, DAA or IBI).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	IBIADDR							0				NOWMAST...	0		
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERRWARN	0	IBIWON	TXNOTFU...	RXPEND	COMPLETE	MCTRLDO...	SLVSTART	IBITYPE		NACKED	BETWEEN	0	STATE		
W			W1C			W1C	W1C	W1C								
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-24 IBIADDR	IBI Address Indicates the address of: • The IBI when MSTATUS[IBITYPE] = 1. • The CR, when MSTATUS[IBITYPE] = 2. • 7'h2 when HJ, when MSTATUS[IBITYPE] = 3.
23-20 —	Reserved
19 NOWMASTER	Module is now Controller Indicates whether the module is now a controller. That is, it was previously a target, and controllership acceptance was requested from the previous controller and controllership was accepted. The reverse operation (controller becomes a target) does not need an interrupt because the application grants it through the GETACCMST CCC.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field behaves differently for register reads and writes. When reading 0b - Not a controller 1b - Controller When writing 0b - No effect 1b - Clear the flag
18-16 —	Reserved
15 ERRWARN	Error or Warning Indicates whether an error occurred, such as improper register use, overrun, or underrun of FIFO or buffer, or invalid parity or CRC in a DDR read. See Controller Errors and Warnings (MERRWARN) for more information. 0b - No error or warning 1b - Error or warning
14 —	Reserved
13 IBIWON	In-Band Interrupt (IBI) Won Indicates whether an IBI, CR, or HJ has won the arbitration on a header address, regardless of whether it was NACKed or ACKed. Arbitration requires manual intervention for CR and HJ, and optionally requires it for IBI if MCTRL[IBIRESP] = 3 (manual). **NOTE** This field behaves differently for register reads and writes. When reading 0b - No IBI arbitration won 1b - IBI arbitration won When writing 0b - No effect 1b - Clear the flag
12 TXNOTFULL	TX Buffer or FIFO Not Full

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates whether the buffer, FIFOm, or message register can accept another byte or halfword. FIFO uses trigger level. If DMA is enabled for transmitting, it transfers data as long as it is not full. 0b - Receive buffer or FIFO full 1b - Receive buffer or FIFO not full
11 RXPEND	RXPEND Indicates whether a message is being received from a target and bytes are in the input buffer or FIFO: • When using a FIFO, this message is at least one FIFO trigger's worth (a minimum of one byte in the FIFO). • When DMA is enabled for receiving, the DMA is signaled. RXPEND becomes 0 when the data is read. 0b - No receive message pending 1b - Receive message pending
10 COMPLETE	Complete Indicates whether a message is complete: • With MWMSG_SDR or MWMSG_DDR, the COMPLETE condition occurs when MWMSG_SDR_CONTROL[LEN] or MWMSG_DDR_CONTROL[LEN] reaches 0. • When MCTRL[REQUEST] = 1 (EmitStartAddr), this COMPLETE condition occurs after the end of a write operation, or after a read operation has terminated or ended. • With an IBI (AutoIBI or EmitStartAddr), the COMPLETE condition occurs at the end of IBI data (if any). **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not complete 1b - Complete When writing 0b - No effect 1b - Clear the flag
9 MCTRLDONE	Controller Control Done Indicates whether the module has completed an MCTRL request. MCTRLDONE automatically becomes 0 when writing a new control. If MCTRL[REQUEST] = 1 (EmitStartAddr), MCTRLDONE becomes 1 when the address goes out (and is ACKed, NACKed, or ended in an IBI). If ACKed, MSTATUS[COMPLETE] becomes 1 after the write or read data is complete.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If MCTRL[REQUEST] = 4 (ProcessDAA), MCTRLDONE becomes 1 when the module is ready to emit the dynamic address (DA) for the target, or when no more targets are ACKing. This condition can be determined by using the MSTATUS[BETWEEN] and MSTATUS[STATE] fields. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not done 1b - Done When writing 0b - No effect 1b - Clear the flag
8 SLVSTART	Target Start Indicates whether a target is or was requesting a START by holding SDA low. Handling starts automatically when MCTRL[REQUEST] = 7 (AutoIBI). **NOTE** This field behaves differently for register reads and writes. When reading 0b - Target not requesting START 1b - Target requesting START When writing 0b - No effect 1b - Clear the flag
7-6 IBITYPE	In-Band Interrupt (IBI) Type Indicates the type of IBI of the last event that won the arbitration, whether the interrupt was ACKed, NACKed, or pending. 00b - NONE (no IBI: this status occurs when MSTATUS[IBIWON] becomes 0) 01b - IBI 10b - CR 11b - HJ
5 NACKED	Not Acknowledged Indicates whether the last start and address sequence was NACKed (was not ACKed by the addressed target). 0b - Not NACKed 1b - NACKed (not acknowledged)

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 BETWEEN	Between Indicates whether the controller is active between messages or dynamic address assignments (DAA). It is active when: • MSTATUS[STATE] is MSGSDR, DDR, or DAA, or NORMACT, and the state is between messages or DAAs. It is expecting a new message or DAA to start (or STOP or exit). • MSTATUS[STATE] is NORMACT. The module is waiting for the transmit FIFO to be not empty or the receive FIFO to be not full. 0b - Inactive (for other cases) 1b - Active
3 —	Reserved
2-0 STATE	State of the Controller Indicates the current controller state. If this field is 0b, it indicates that the controller is idle (the bus has stopped.) If this field is 1b, it indicates target request. The bus has stopped but a target is holding SDA low. When using auto-emit IBI (MCTRL[REQUEST] = 7), the controller does not remain in this state. If this field is 10b, it indicates entry into Single Data Rate Message mode, using MWMSG_SDR. If this field is 11b, it indicates entry into normal active SDR mode, using MCTRL, MWDATA*n*, and MRDATA*n* registers. The controller remains in this state until a STOP is issued. If this field is 100b, it indicates entry into Double Data Rate Message mode, using MWMSG_DDR or the normal method with DDR. The controller remains in the DDR state until the controller exits using EXIT (emitting the Exit pattern). If this field is 101b, it indicates entry into Dynamic Address Assignment (ENTDAA) mode. If this field is 110b, it indicates wait for an IBI ACK/NACK decision. If this field is 111b, it indicates receiving an IBI. This state is used after IBI, CR, or HJ has won an arbitration. The IBIRCV state is also used for IBI mandatory byte (if any) and any bytes that follow. 000b - IDLE (bus has stopped) 001b - SLVREQ (target request) 010b - MSGSDR 011b - NORMACT 100b - MSGDDR 101b - DAA 110b - IBIACK 111b - IBIRCV

72.7.30 Controller In-band Interrupt Registry and Rules (MIBIRULES)

Offset

Register	Offset
MIBIRULES	8Ch

Function

Contains the rules for using IBI, and keeps a registry of the targets that use the IBI byte.

This register defines the IBI mandatory byte rules for the targets and determines which targets do (or do not) have a mandatory byte.

Concerning ADDR*n* fields: The address has 6 bits. Assuming that MIBIRULES[MSB0] = 1, the most significant bit of each address is 0. In this case, each address is 7 bits (usually written as A7 to A1) and A7 must be 0. If the application does not use that optimal convention, the "manual" method of IBI ACK handling must be used (see MCTRL[IBIRESP] for more information).

By default, the ADDR*n* values indicate the targets with a mandatory byte. If MIBIRULES[NOBYTE] = 1, the ADDR*n* values indicate targets that do not have a mandatory IBI byte.

NOTE

A7 = 0 is only needed for targets that use IBI. For legacy I2C devices, A7 can use any valid value because I2C devices cannot cause an IBI.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NOBYTE	MSB0	ADDR4						ADDR3						ADDR2	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR2				ADDR1				ADDR0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 NOBYTE	No IBI byte Specifies whether the ADDR*n* fields refer to targets with or without a mandatory IBI byte. 0b - With mandatory IBI byte 1b - Without mandatory IBI byte
30 MSB0	Most Significant Address Bit is 0

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Specifies whether MSB is 0 for all I3C dynamic addresses. Assigning 1 to this field allows the START header to be optimized. 0b - MSB is not 0 1b - MSB is 0
29-24 ADDR4	ADDR4 Specifies the address of the target with or without mandatory IBI byte. If 0, then the address does not apply.
23-18 ADDR3	ADDR3 Specifies the address of the target with or without mandatory IBI byte. If 0, then the address does not apply.
17-12 ADDR2	ADDR2 Specifies the address of the target with or without mandatory IBI byte. If 0, then the address does not apply.
11-6 ADDR1	ADDR1 Specifies the address of the target with or without mandatory IBI byte. If 0, then the address does not apply.
5-0 ADDR0	ADDR0 Specifies the address of the target with or without mandatory IBI byte. If 0, then the address does not apply.

72.7.31 Controller Interrupt Set (MINTSET)

Offset

Register	Offset
MINTSET	90h

Function

Returns the status of the interrupt enables when read. This register sets interrupt enables for select fields in Controller Status (MSTATUS):

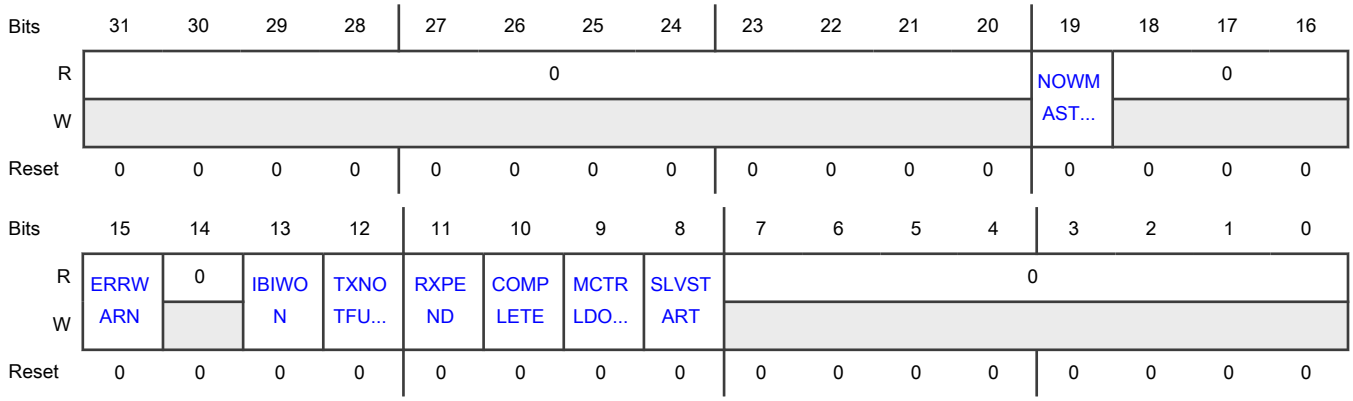
- To activate an interrupt enable, write 1 to its corresponding field.
- To disable an interrupt enable, write 1 to the appropriate field in Controller Interrupt Clear (MINTCLR). Writing 0 to the interrupt enable in MINTSET does not disable the interrupt.

The interrupt registers allow masking of interrupt sources as well as checking which interrupts are activated in Controller Status (MSTATUS). The normal method is to enable an interrupt and then after that interrupt fires, clear the interrupt either by writing to Controller Status (MSTATUS) or via an action on the corresponding data register. The interrupt is level-held, meaning that the interrupt remains 1 until the cause is cleared by some method. The module prevents races; if a new event comes in, that new event is not lost.

These interrupts are parallel to Target Interrupt Set (SINTSET) and Target Interrupt Clear (SINTCLR), and only one interrupt set is active depending on the state (controller or target operation):

- MINTSET: Sets interrupt enables for MSTATUS fields. Reading the MINTSET register returns the status of the interrupt enables.
- MINTCLR: Clears interrupt enables for MSTATUS fields.
- MINTMASKED: Returns the value of the MSTATUS fields ANDed with their interrupt enables.

Diagram



Fields

Field	Function
31-20 —	Reserved
19 NOWMASTER	Now Controller Interrupt Enable Enables the corresponding interrupt to indicate whether the module is now a controller (now this I3C module is a controller). That is, it was previously a target; controllership acceptance was requested from the previous controller and it was accepted. 0b - Disable 1b - Enable
18-16 —	Reserved
15 ERRWARN	Error or Warning (ERRWARN) Interrupt Enable Enables the corresponding interrupt to indicate whether an error occurred, such as improper register use, overrun or underrun of FIFO or buffer, or invalid parity or CRC in a DDR read. 0b - Disable 1b - Enable
14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 IBIWON	IBI Won Interrupt Enable Enables the corresponding interrupt to indicate whether an IBI, CR, or HJ has won the arbitration on a header address, regardless of whether it was NACKed or ACKed. 0b - Disable 1b - Enable
12 TXNOTFULL	Transmit Buffer/FIFO Not Full Interrupt Enable Enables the corresponding interrupt to indicate whether the buffer, FIFOm, or message register can accept another byte or halfword. 0b - Disable 1b - Enable
11 RXPEND	Receive Pending Interrupt Enable Enables the corresponding interrupt to indicate whether a message is being received from a target and bytes are in the input buffer or FIFO.
10 COMPLETE	Completed Message Interrupt Enable Enables the corresponding interrupt to indicate whether a message is complete. 0b - Disable 1b - Enable
9 MCTRLDONE	Controller Control Done Interrupt Enable Enables the corresponding interrupt to indicate whether the module has completed an MCTRL request. 0b - Disable 1b - Enable
8 SLVSTART	Target Start Interrupt Enable Enables the corresponding interrupt to indicate whether a target is or was requesting a START by holding SDA low. 0b - Disable 1b - Enable
7-0 —	Reserved

72.7.32 Controller Interrupt Clear (MINTCLR)

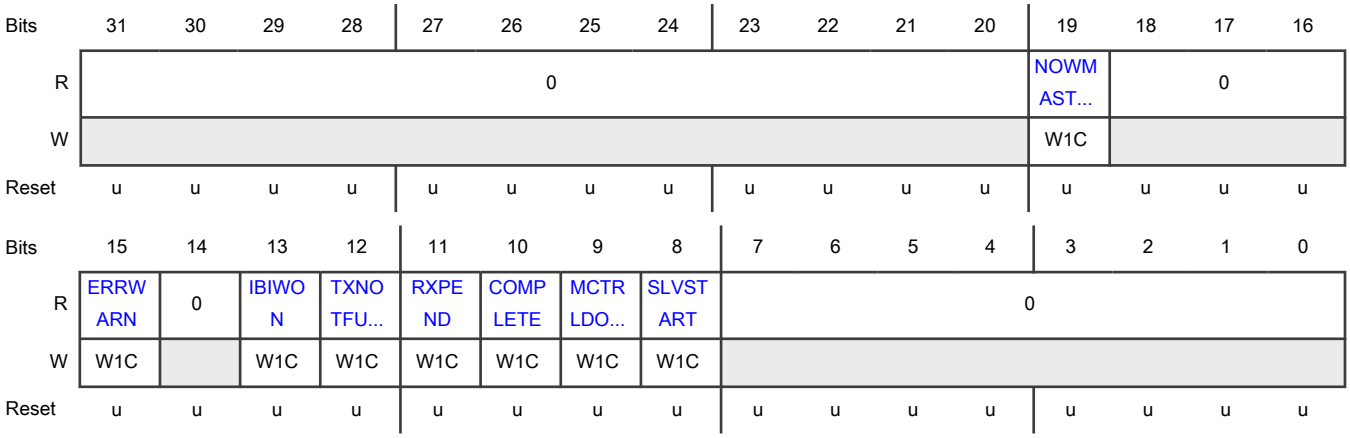
Offset

Register	Offset
MINTCLR	94h

Function

Clears interrupt enables for select fields in Controller Status (MSTATUS). Writing 1 clears the corresponding interrupt enable. Writing 0 has no effect.

Diagram



Fields

Field	Function
31-20 —	Reserved
19 NOWMASTER	NOWCONTROLLER Interrupt Enable Clear Clears the corresponding NOWCONTROLLER interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing 0b - No effect 1b - Clear the flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-16 —	Reserved
15 ERRWARN	ERRWARN Interrupt Enable Clear Clears the corresponding ERRWARN interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing 0b - No effect 1b - Clear the flag
14 —	Reserved
13 IBIWON	IBIWON Interrupt Enable Clear Clears the corresponding IBIWON interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing 0b - No effect 1b - Clear the flag
12 TXNOTFULL	TXNOTFULL Interrupt Enable Clear Clears the corresponding TXNOTFULL interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Clear the flag
11 RXPEND	RXPEND Interrupt Enable Clear Clears the corresponding RXPEND interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing 0b - No effect 1b - Clear the flag
10 COMPLETE	COMPLETE Interrupt Enable Clear Clears the corresponding COMPLETE interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing 0b - No effect 1b - Clear the flag
9 MCTRLDONE	MCTRLDONE Interrupt Enable Clear Clears the corresponding MCTRLDONE interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing 0b - No effect 1b - Clear the flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 SLVSTART	SLVSTART Interrupt Enable Clear Clears the corresponding SLVSTART interrupt enable. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No effect 1b - Interrupt enable cleared When writing 0b - No effect 1b - Clear the flag
7-0 —	Reserved

72.7.33 Controller Interrupt Mask (MINTMASKED)

Offset

Register	Offset
MINTMASKED	98h

Function

Returns the status of enabled interrupts (the value of Controller Status (MSTATUS) ANDed with the value of Controller Interrupt Set (MINTSET)).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												NOWMAST...	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERRWARN	0	IBIWON	TXNOTFU...	RXPEND	COMPLETE	MCTRLDO...	SLVSTART	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19 NOWMASTER	NOWCONTROLLER Interrupt Mask Indicates whether the NOWCONTROLLER interrupt is enabled and active. 0b - Disabled 1b - Enabled
18-16 —	Reserved
15 ERRWARN	ERRWARN Interrupt Mask Indicates whether the ERRWARN interrupt is enabled and active. 0b - Disabled 1b - Enabled
14 —	Reserved
13 IBIWON	IBIWON Interrupt Mask Indicates whether the IBIWON interrupt is enabled and active. 0b - Disabled 1b - Enabled
12 TXNOTFULL	TXNOTFULL Interrupt Mask Indicates whether the TXNOTFULL interrupt is enabled and active. 0b - Disabled 1b - Enabled
11 RXPEND	RXPEND Interrupt Mask Indicates whether the RXPEND interrupt is enabled and active.
10 COMPLETE	COMPLETE Interrupt Mask Indicates whether the COMPLETE interrupt is enabled and active. 0b - Disabled 1b - Enabled
9 MCTRLDONE	MCTRLDONE Interrupt Mask Indicates whether the MCTRLDONE interrupt is enabled and active.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled 1b - Enabled
8 SLVSTART	SLVSTART Interrupt Mask Indicates whether the SLVSTART interrupt is enabled and active. 0b - Disabled 1b - Enabled
7-0 —	Reserved

72.7.34 Controller Errors and Warnings (MERRWARN)

Offset

Register	Offset
MERRWARN	9Ch

Function

Contains errors and warnings. When any errors or warnings are not 0, then MSTATUS[ERRWARN] is 1.

Parallel-to-target ERRWARN:

- In Controller mode, use this register (MERRWARN).
- In Target mode, use Target Errors and Warnings (SERRWARN).

The error fields in both registers (MERRWARN and SERRWARN) are similar in meaning.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										0	TIMEOUT	INVREQ	MSGERR	OWRITE	OREAD
W												W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			0	0	HCRC	HPAR	0				Reserved	WRABT	NACK	0	0
W						W1C	W1C					W1C	W1C	W1C		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-22 —	Reserved
21 —	Reserved
20 TIMEOUT	Timeout Error Indicates an error caused by the module stalling for too long in a frame. This stalling occurs when: • The transmit FIFO or receive FIFO is not handled, and the bus is stuck in the middle of a message. • No STOP is issued after data transfer (there is a gap between messages). • IBI manual is used and no decision has been made. The maximum stall period is 10 kHz or 100 µs. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag
19 INVREQ	Invalid Request Error Indicates an error caused by an invalid use of a request: • Not using IBI ACK NACK when stopped in manual hold for IBI acknowledgment. • Using a request other than ForceStop or ForceExit when in a message. Other requests are valid when the message is done. • Other mismatched uses (for example, IBI ACK NACK in normal states). **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 MSGERR	Message Error Indicates an error caused by: • Trying to write to or read from the MWMSG_SDR register when in a DDR message. • Trying to write to or read from the MWMSG_DDR register when in an SDR message. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag
17 OWRITE	Overwrite Error Indicates an error caused by trying to write to Controller Write Data Byte (MWDATAB) when the FIFO is full. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag
16 OREAD	Overread Error Indicates an error caused by: • Trying to read from Controller Read Data Byte (MRDATAB) when the FIFO is empty. • Trying to read from the MRMSG_SDR register or the MRMSG_DDR register when no message has yet started. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Error When writing 0b - No effect 1b - Clear the flag
15-13 —	Reserved
12 —	Reserved
11 —	Reserved
10 HCRC	High Data Rate CRC Error Indicates that a cyclic redundancy check (CRC) error occurred from a DDR read. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag
9 HPAR	High Data Rate Parity Indicates a parity error from a DDR read, including a bad preamble on a read. This does not stop the read because it is not safe to terminate; the read data may become misframed. Ends on a run of 1 second. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
8-5 —	Reserved
4 —	Reserved
3 WRABT	Write Abort Error Indicates an error caused by the I2C target NACKing the write data, terminating the message. For example, the controller is writing in I2C and the target NACKed the write. If you write to the MCTRL register, this field automatically becomes 0. **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag
2 NACK	Not Acknowledge Error Indicates an error caused by the target or targets NACKing (not acknowledging) the last address. If 7Eh is the address, then it indicates all targets NACKed the last address. If you write to the MCTRL register, this field automatically becomes 0. **NOTE** In HDR mode, this error occurs when an address is not accepted (as opposed to NACKed). **NOTE** This field behaves differently for register reads and writes. When reading 0b - No error 1b - Error When writing 0b - No effect 1b - Clear the flag
1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 —	Reserved

72.7.35 Controller DMA Control (MDMACTRL)

Offset

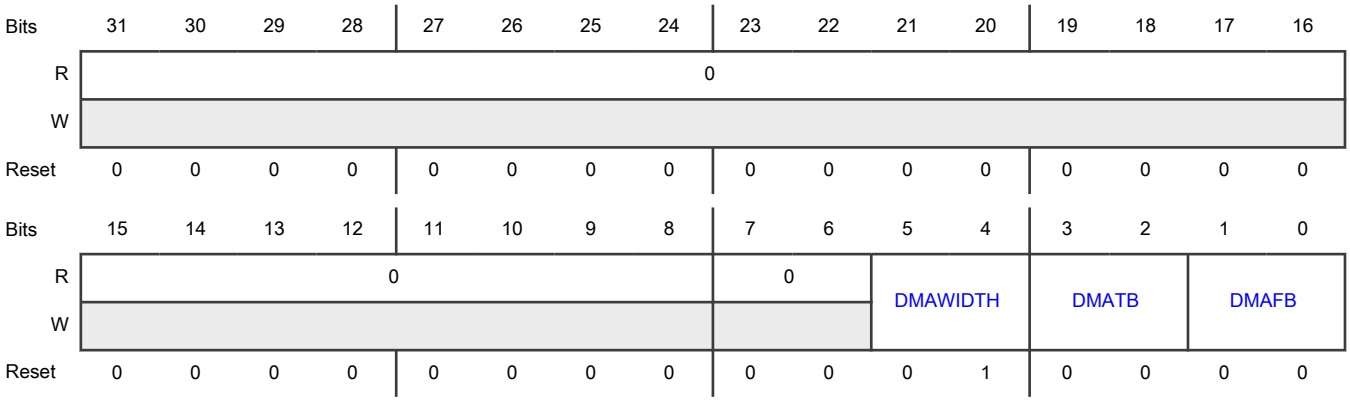
Register	Offset
MDMACTRL	A0h

Function

DMA can be used with a controller in these ways:

- Push or pull data for MCTRL[REQUEST] = 1 (EmitStartAddr) request written by the processor.
- Implementing message mode, completely controlled by DMA.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-6 —	Reserved
5-4 DMAWIDTH	DMA Width Specifies the data width of DMA operations.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The halfword (16 bits) setting ensures that two bytes are free or available in FIFO. 00b,01b - Byte 10b - Halfword (16 bits) 11b - Reserved
3-2 DMATB	DMA to Bus Represents the DMA write (to-bus) trigger. When DMAFB = 1 or DMAFB = 2, I3C requests DMA when a transmit trigger occurs (see Controller Data Control (MDATACTRL)). I3C requests until full, unless the DMA is set up as a trigger. DMAFB becomes 0 when MSTATUS[ERRWARN] = 1. If this field is 1b, STOP or START causes DMATB to become 0 (see SCONFIG[MATCHSS]). Normally, DMA enable must be used only in Controller Message mode. 00b - DMA not used 01b - Enable DMA for one frame (ended by DMA or terminated) 10b - Enable DMA until DMA is turned off 11b - Reserved
1-0 DMAFB	DMA from Bus Represents the DMA read (from-bus) trigger. When DMAFB = 1 or DMAFB = 2, I3C requests DMA when a receive trigger occurs (see Controller Data Control (MDATACTRL)). I3C requests until empty unless the DMA is set up as a trigger. DMAFB becomes 0 when MSTATUS[ERRWARN] = 1. If this field is 1b, STOP or Repeated START causes DMAFB to automatically become 0 (see SCONFIG[MATCHSS]). **NOTE** Do not enable DMA after a transaction starts. It must be enabled only when enabling the controller and interrupts to avoid any kind of RX FIFO overrun. 00b - DMA not used 01b - Enable DMA for one frame 10b - Enable DMA until DMA is turned off 11b - Reserved

72.7.36 Controller Data Control (MDATACTRL)

Offset

Register	Offset
MDATACTRL	ACh

Function

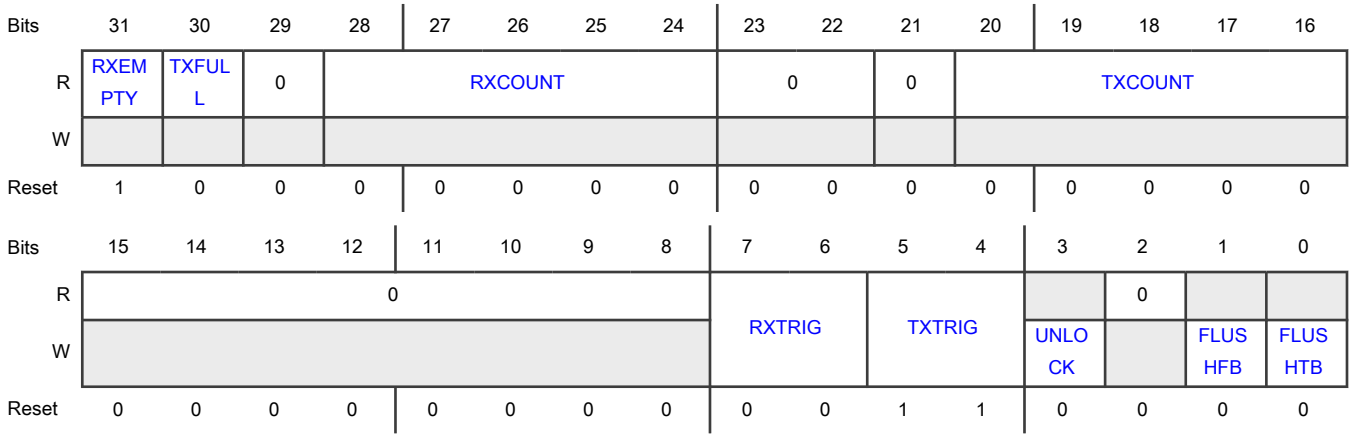
Assists in data control when there is no FIFO. This register also assists the FIFO when FIFO is available (regardless of size), and this assistance allows some control over the FIFO behavior. In particular, the register provides control over when to interrupt when a particular state of fullness or emptiness is reached. It also controls behavior related to width, when the width is not 1 byte wide.

This register acts as an alias of Target Data Control (SDATACTRL).

NOTE

When flushing a FIFO if DMA is in use, disable the DMA channel first. You must not use FIFO flush when a message (in that direction) is in flight.

Diagram



Fields

Field	Function
31 RXEMPTY	Receive is Empty Indicates whether the receive FIFO or buffer is empty. 0b - Not empty 1b - Empty
30 TXFULL	Transmit is Full Indicates whether the transmit FIFO or buffer is full. 0b - Not full 1b - Full
29 —	Reserved
28-24 RXCOUNT	Receive Byte Count Contains the count of bytes in the receive FIFO or buffer.
23-22	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
21 —	Reserved
20-16 TXCOUNT	Transmit Byte Count Contains the count of bytes waiting in the TXFIFO. This count is the number of bytes that the application has written to the transmit FIFO that have not yet gone onto the bus.
15-8 —	Reserved
7-6 RXTRIG	Receive Trigger Level Indicates the trigger level for receive fullness when using a FIFO. This field affects the RXPEND interrupt. 00b - Trigger when not empty 01b - Trigger when 1/4 full or more 10b - Trigger when 1/2 full or more 11b - Trigger when 3/4 full or more
5-4 TXTRIG	Transmit Trigger Level Indicates the trigger level for transmit emptiness when using a FIFO. This field affects the TXNOTFULL interrupt. 00b - Trigger when empty 01b - Trigger when 1/4 full or less 10b - Trigger when 1/2 full or less 11b - Trigger when 1 less than full or less (default)
3 UNLOCK	Unlock Unlocks the functionality so that the RXTRIG and TXTRIG fields can be changed on a write. This field must be 1 (unlocked) in the same cycle when writing to TXTRIG or RXTRIG. If this field is 0 (locked), RXTRIG and TXTRIG fields cannot be changed on a write. 0b - Locked 1b - Unlocked
2 —	Reserved
1 FLUSHFB	Flush From-Bus Buffer or FIFO Flushes from-bus buffer or FIFO. You normally do not use this field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No action 1b - Flush the buffer
0 FLUSHTB	Flush To-Bus Buffer or FIFO Flushes to-bus buffer or FIFO. This field is used when the controller terminates a to-bus message (read) prematurely. 0b - No action 1b - Flush the buffer

72.7.37 Controller Write Data Byte (MWDATAB)

Offset

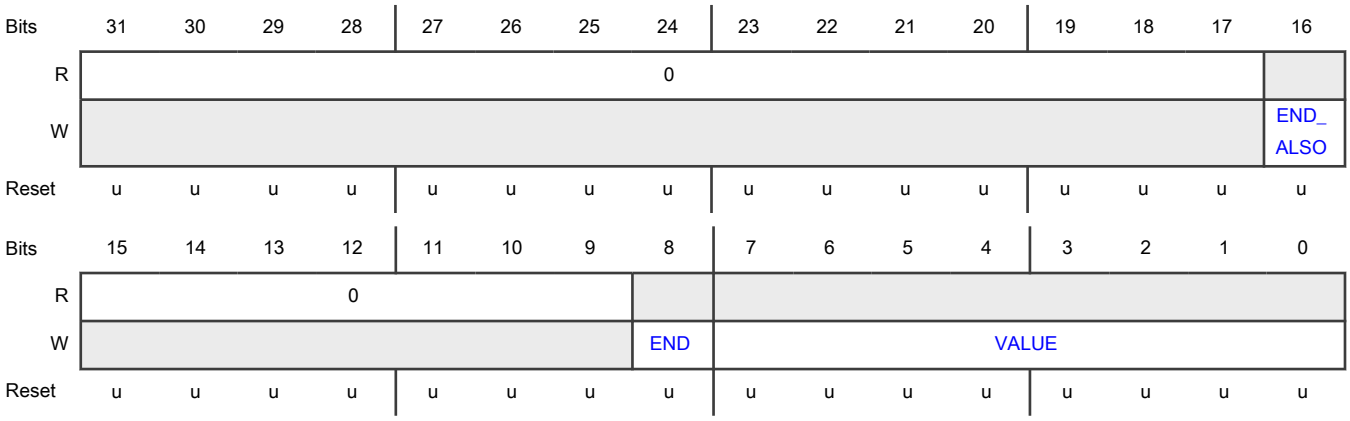
Register	Offset
MWDATAB	B0h

Function

Allows writing bytes to send onto the bus. This register is used only when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

This register acts as an alias of Target Write Data Byte (SWDATAB).

Diagram



Fields

Field	Function
31-17	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
16 END_ALSO	End of Message ALSO Indicates end of message, used to end an outbound message normally. Every message must indicate when it is the last message to be sent. This method can be used with the MDATABE register. This field is required for I3C and is optional for I2C. For HDR-DDR, the byte with END_ALSO must be an even byte (second, fourth, sixth, and so on) because DDR uses byte pairs. If this field is 0, more bytes are assumed to be in the message. If this field is 1, the END bit marks the last byte of the message. 0b - Not the end 1b - End
15-9 —	Reserved
8 END	End of Message Indicates the end of a message. This field is required for I3C and is optional for I2C. For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte pairs. If this field is 0, more bytes are assumed to be in the message. If this field is 1, the END bit marks the last byte of the message. 0b - Not the end 1b - End
7-0 VALUE	Data Byte Represents the byte written to the target (stored in transmit FIFO): • I3C computes the parity. • I2C manages the ACK or NACK.

72.7.38 Controller Write Data Byte End (MWDATABE)

Offset

Register	Offset
MWDATABE	B4h

Function

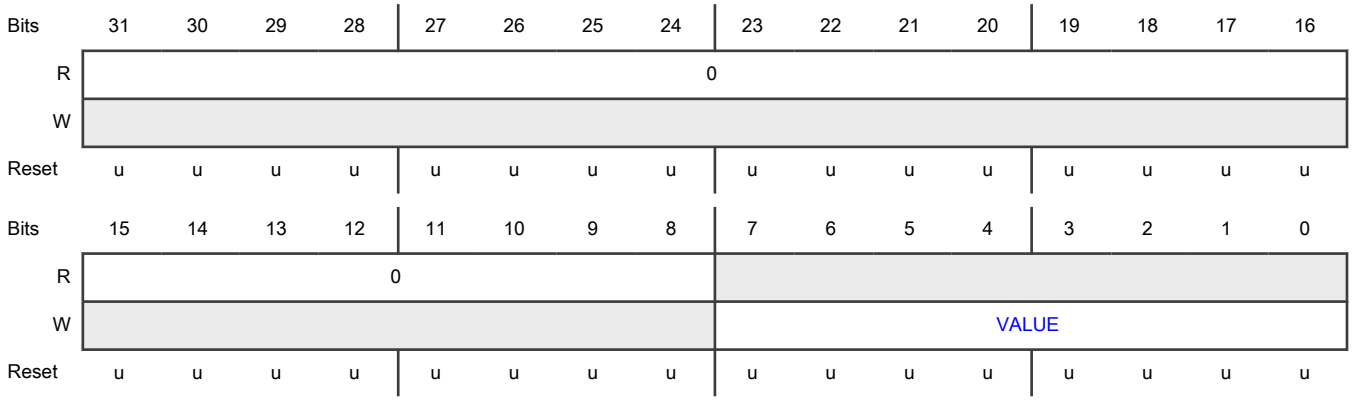
Allows writing the last byte to send onto the bus. This register is used only when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

This register acts as an alias of Target Write Data Byte End (SWDATABE).

NOTE

MWDATABE can also indicate END using bit 8.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 VALUE	Data Represents the last byte written to the target (stored in transmit FIFO): • I3C computes the parity. • I2C manages the ACK or NACK.

72.7.39 Controller Write Data Halfword (MWDATAH)

Offset

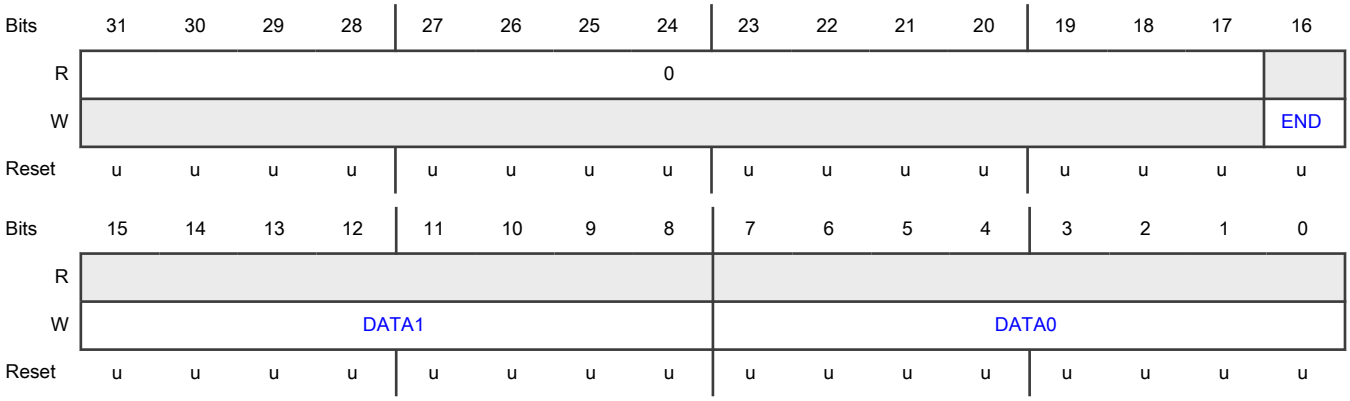
Register	Offset
MWDATAH	B8h

Function

Allows writing a halfword (pair of bytes) to the bus unless an external FIFO is used, sending the low byte followed by the high byte. An end-of-data (last) marker bit is allowed (or must be 0). A halfword must not be written unless there is room for both, as indicated by the use of transmit FIFO level trigger or MDATACTRL[TXCOUNT].

This register acts as an alias of Target Write Data Halfword (SWDATAH).

Diagram



Fields

Field	Function
31-17 —	Reserved
16 END	End of Message Indicates the end of a message. For this register, this field always marks DATA1 as the end. This field is required for I3C and is optional for I2C. For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte pairs. If this field is 0, more bytes are assumed to be in the message. If this field is 1, the END bit marks the last byte of the message. 0b - Not the end 1b - End
15-8 DATA1	Data Byte 1 Represents the second byte that is sent to the target (written to transmit FIFO).
7-0 DATA0	Data Byte 0 Represents the first byte that is sent to the target (written to transmit FIFO).

72.7.40 Controller Write Data Halfword End (MWDATAHE)

Offset

Register	Offset
MWDATAHE	BCh

Function

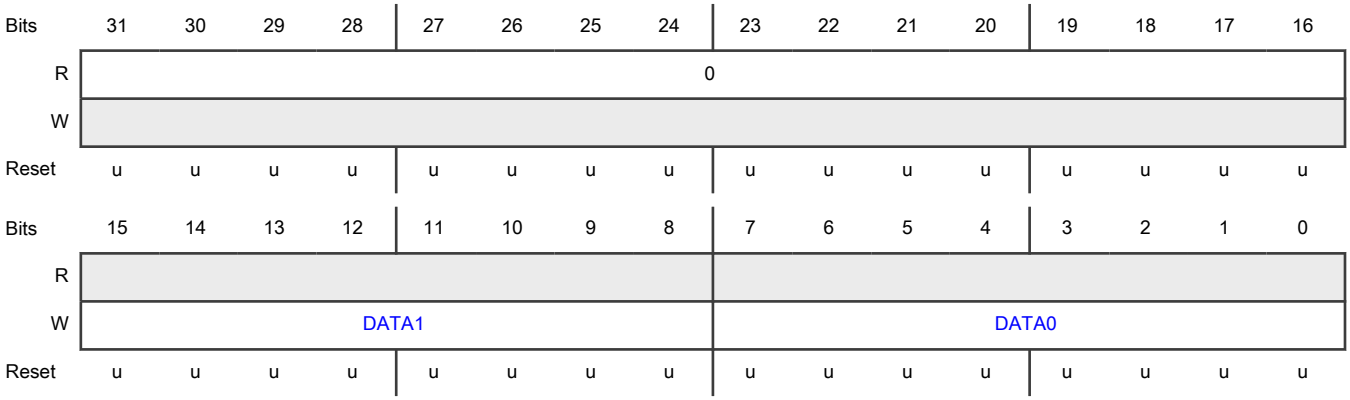
Allows writing a halfword of data, which is the end (the last byte of the halfword is the end). The target writes the halfword (byte pair) in the same way as it does to Controller Write Data Halfword (MWDATAH), but marks the second byte as end-of-data (last byte).

For HDR-DDR, the byte with the END must be even (second, fourth, sixth, and so on) because DDR uses byte pairs.

You must not write a halfword unless there is room for both, as indicated by the use of transmit FIFO level trigger or MDATACTRL[TXCOUNT].

This register acts as an alias of Target Write Data Halfword End (SWDATAHE).

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 DATA1	Data Byte 1 Represents the second byte that is sent to the target (written to transmit FIFO).
7-0 DATA0	Data Byte 0 Represents the first byte that is sent to the target (written to transmit FIFO).

72.7.41 Controller Read Data Byte (MRDATAB)

Offset

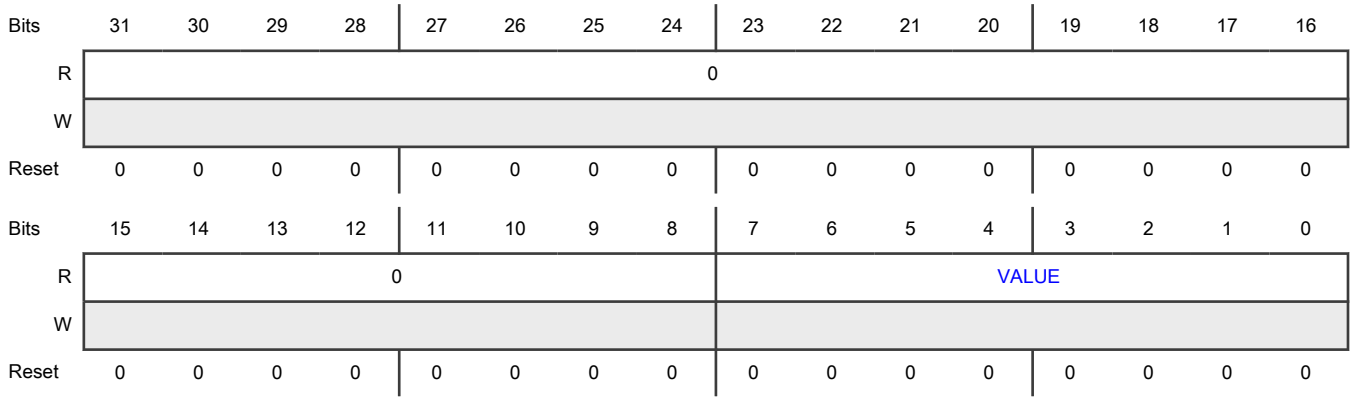
Register	Offset
MRDATAB	C0h

Function

Allows reading bytes written by the target after an SDR read, or DAA or DDR. This register is used only when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

This register acts as an alias of Target Read Data Byte (SRDATAB).

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 VALUE	Value Represents the byte read from the controller (and written by the target).

72.7.42 Controller Read Data Halfword (MRDATAH)

Offset

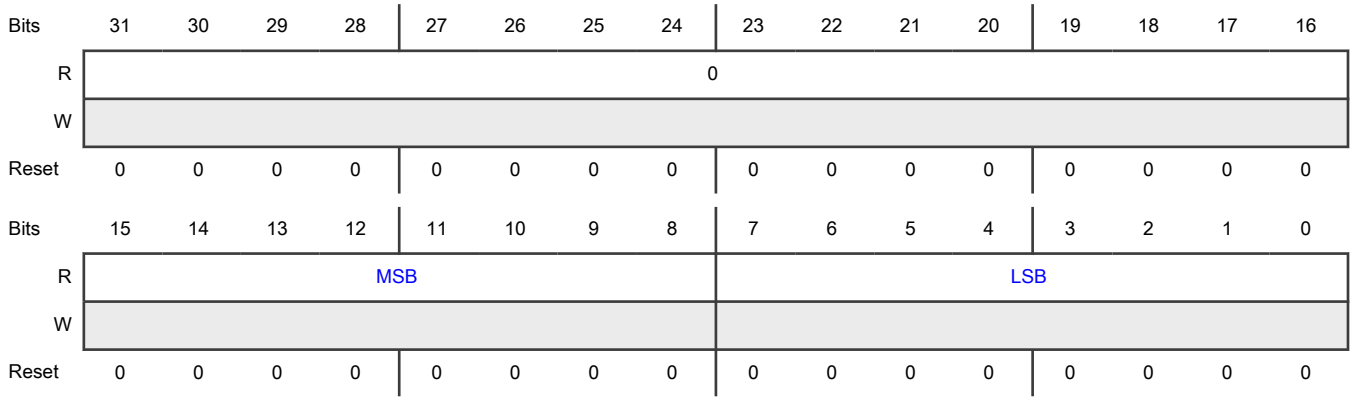
Register	Offset
MRDATAH	C8h

Function

Allows reading a halfword (byte pair) written by the target after an SDR read or DAA or DDR. This register is used only when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

A halfword must not be read unless there are at least two bytes of data waiting, as indicated by the receive FIFO level trigger or MDATACTRL[RXCOUNT].

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MSB	High Byte Represents the second byte read from the controller (and written by the target).
7-0 LSB	Low Byte Represents the first byte read from the controller (and written by the target).

72.7.43 Controller Write Byte Data 1 (to Bus) (MWDATAB1)

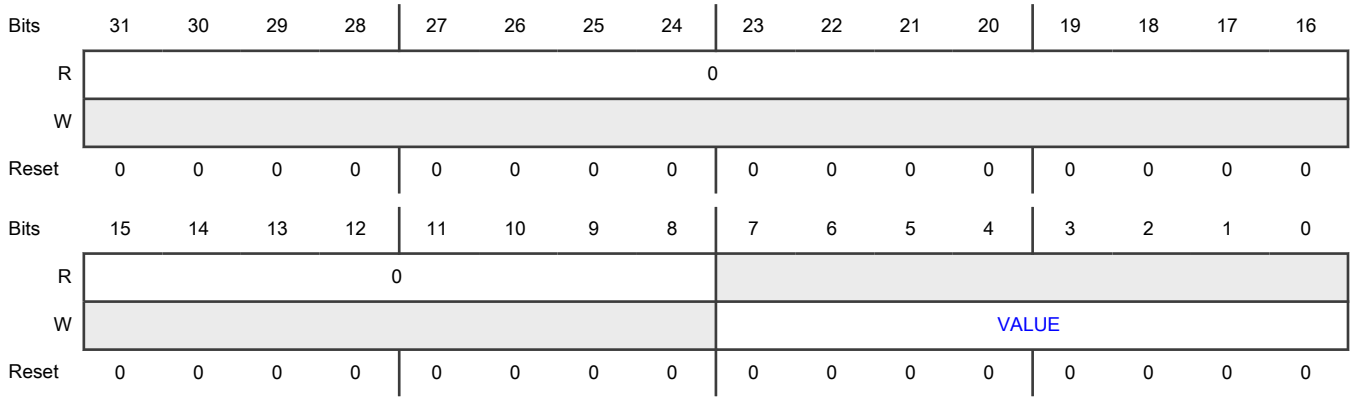
Offset

Register	Offset
MWDATAB1	CCh

Function

Allows writing bytes to send onto the bus, and is intended for DMAs that do not clear the upper bits of the word. This register does not have the END bits and is only used when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 VALUE	Value Represents the byte to write out. The I3C module computes the parity for I3C, or manages the ACK or NACK for I2C.

72.7.44 Controller Write Message Control in SDR mode (MWMSG_SDR_CONTROL)

Offset

Register	Offset
MWMSG_SDR_CONTROL	D0h

Function

Allows setting up and writing 16-bit words in Single Data Rate (SDR) mode. The MWMSG_SDR_ register is modal and has two modes—control and data:

- For the first write to set up a new message, this register functions as the MWSMSG_SDR_CONTROL register.
- For subsequent writes, this register functions as MWMSG_SDR_DATA.
- The control information is not pipelined. You must write to control information registers for a start and do not write to them until data is sent.

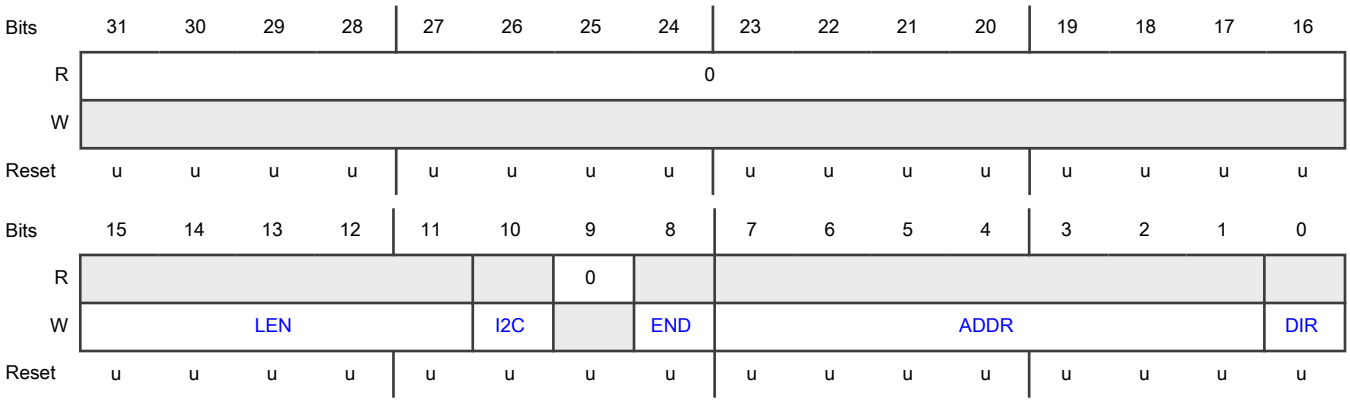
When not in the middle of a message, the MWMSG_SDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_SDR_DATA register is used until the length (see the LEN field) counts down, or until data with END = 1 is used.

The MWMSG_SDR_CONTROL register is written with the 16-bit control word if currently stopped or after the end of the previous message. If MSTATUS[STATE] = 2 (MSGSDR) and MSTATUS[BETWEEN] = 0, the register (at this offset address) functions as the MWMSG_SDR_DATA register. Otherwise, this register (at this offset address) functions as the MWMSG_SDR_CONTRL register, as long as MSTATUS[STATE] is not in another mode.

The control word contains the byte length (6-bit), address, direction, and how it ends (stop, ready for next, continuation with more length). If the command is START and an event (IBI, CR, HJ) occurs, MCTRL[IBIRESP] is used to determine action, and the corresponding interrupt occurs. In that case, the message is restarted.

The MWMSG_SDR_CONTROL and MWMSG_SDR_DATA registers are only targeted for DMA operations, but the processor can also write to the MWMSG_SDR_CONTROL register (instead of using MCTRL and MxDATAB).

Diagram



Fields

Field	Function
31-16 —	Reserved
15-11 LEN	Length Indicates the byte length of the message. If LEN = 0, then only END is used (and it does not use the address if stopped). LEN = 1 must not be used. The minimal LEN size is 2 bytes (LEN = 2).
10 I2C	I2C Specifies whether the message is I2C or I3C. 0b - I3C message 1b - I2C message
9 —	Reserved
8 END	End of SDR Message Indicates the end of SDR message. MSTATUS[COMPLETE] = 1 when done. The end can happen: • Before LEN bytes are read in I3C mode, if a target ends sooner. • Before LEN bytes are written in I2C mode, if a target NACKs. If this field is 0, the SDR message ends waiting for a new SDR message (issues a Repeated START for a new message). If this field is 1, the SDR message ends at the STOP.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not the end 1b - End
7-1 ADDR	Address Contains the address to be written.
0 DIR	Direction 0b - Write 1b - Read

72.7.45 Controller Write Message Data in SDR mode (MWMSG_SDR_DATA)

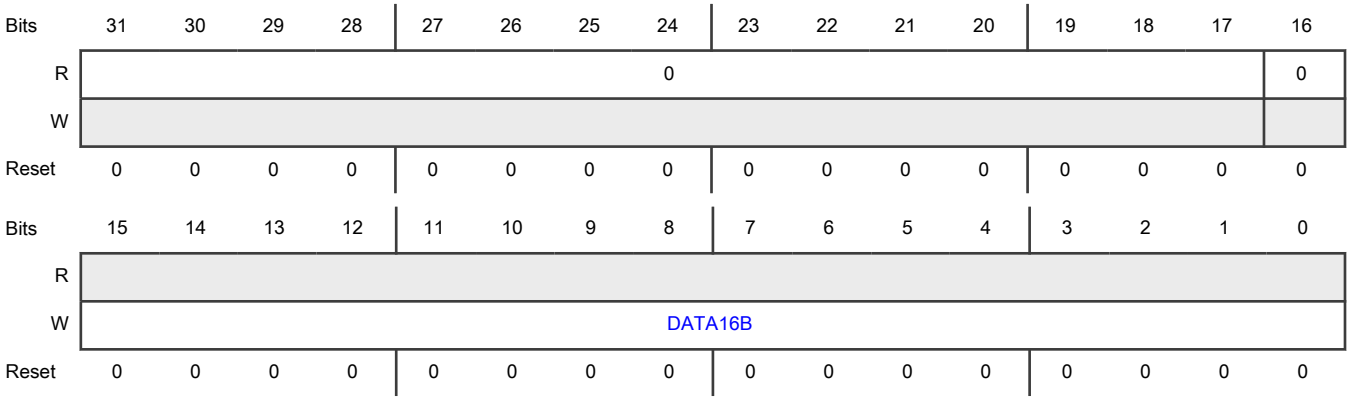
Offset

Register	Offset
MWMSG_SDR_DATA	D0h

Function

Contains the 16-bit word to be written in Single Data Rate (SDR) mode. This register functions in a way similar to MWMSG_SDR_CONTROL.

Diagram



Fields

Field	Function
31-17 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 —	Reserved
15-0 DATA16B	Data

72.7.46 Controller Read Message in SDR mode (MRMSG_SDR)

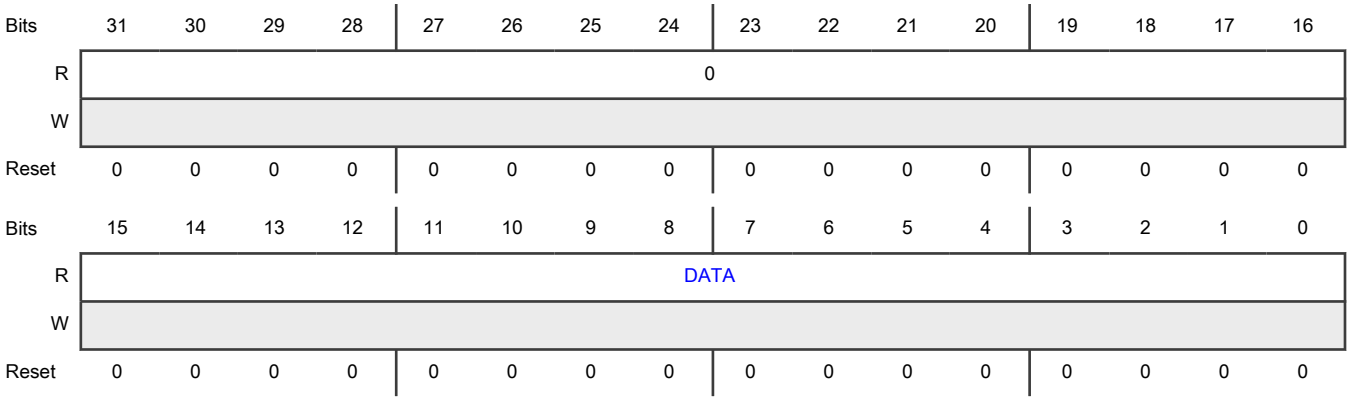
Offset

Register	Offset
MRMSG_SDR	D4h

Function

Allows reading 16-bit words from a target in SDR Message mode. The MRMSG_SDR register is used to read 16-bit words from an active message started with MWMSG_SDR. These words are intended to be read by DMA.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 DATA	Data Contains the 16-bit word read from the target: • If the length (LEN) is an odd number, the upper byte is 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• If the target ends before LEN is finished, the module treats the read as completed. • If the target is not done before LEN is finished and END is not a continuation, the read is terminated.

72.7.47 Controller Write Message in DDR mode: First Control Word (MWMSG_DDR_CONTROL)

Offset

Register	Offset
MWMSG_DDR_CONTROL	D8h

Function

Allows setting up and writing 16-bit words in DDR mode. The register has three modes:

- The first write to set up a new message functions as the MWSMSG_DDR_CONTROL register.
- The second write contains the functions as shown in the MWMSG_DDR_CONTROL2 register.
- For subsequent writes, this register functions in a way similar to Controller Write Message Data in DDR mode (MWMSG_DDR_DATA).
- The control information is not pipelined. You must write to control information registers for a start and do not write to them until data is sent.

When not in the middle of a message, the MWMSG_DDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_DDR_DATA register is used until the length (see LEN field) counts down or until data with END = 1 is used.

The MWMSG_DDR_CONTROL register is written with the 16-bit control word if currently stopped or after the end of the previous message. If MSTATUS[STATE] = 4 (MSGDDR) and MSTATUS[BETWEEN] = 0, then the register (at this offset address) functions as the MWMSG_DDR_DATA register. Otherwise, this register (at this offset address) functions as the MWMSG_DDR_CONTROL register, as long as MSTATUS[STATE] is not in another mode.

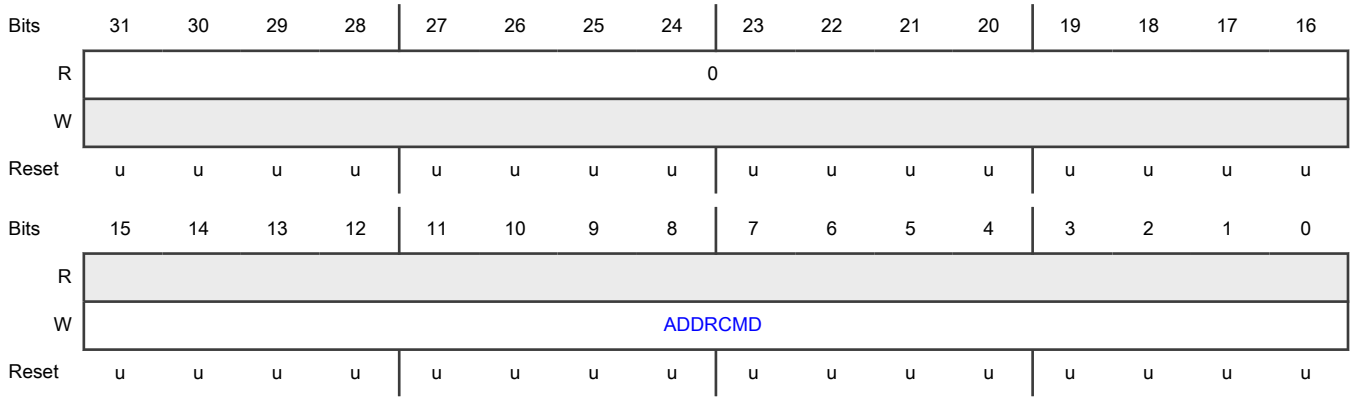
The main control word contains the 16-bit word length and how it ends (stop, ready for next, continuation with more length). Then the command word contains the command and address for read or write. If the command is START and an event (IBI, CR, HJ) occurs, MCTRL[IBIRESP] is used to determine action, and the corresponding interrupt occurs. In that case, the message is restarted.

The MWMSG_DDR_CONTROL, MWMSG_DDR_CONTROL2, and MWMSG_DDR_DATA registers are only targeted for DMA operations, but the processor can also write to the MWMSG_DDR_CONTROL register (instead of using MCTRL and MxDATAB).

NOTE

The module handles preamble, parity, and CRC.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 ADDRCMD	Address Command Indicates the first data write after control write, with LEN ≠ 0. This is formatted as: • Bits 15:9: target address to read or write • Bit 8: reserved, must be 0 • Bit 7: 1 if read, 0 if write • Bits 6:0: CMD as 7-bit value, always for controller

72.7.48 Controller Write Message in DDR Mode Control 2 (MWMSG_DDR_CONTROL2)

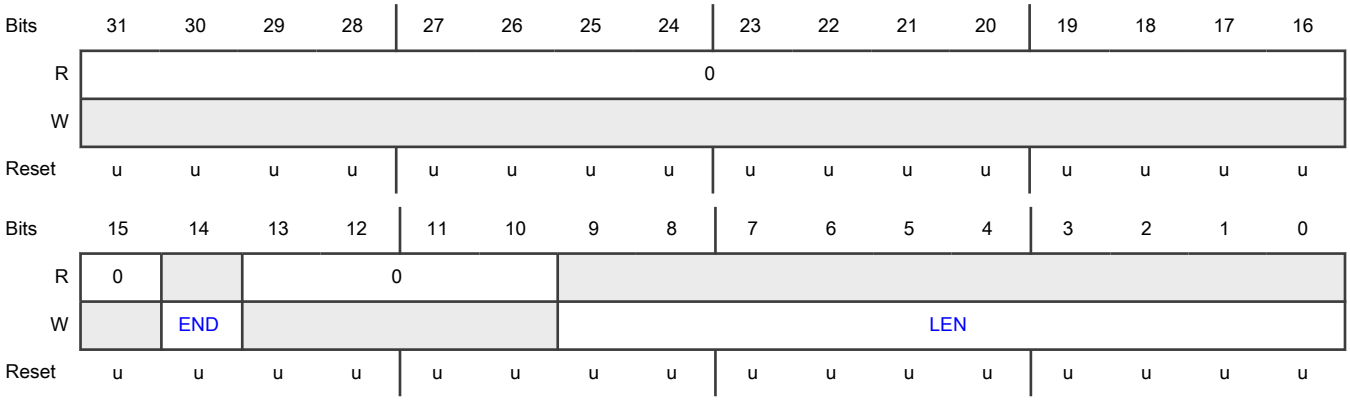
Offset

Register	Offset
MWMSG_DDR_CONTROL2	D8h

Function

Contains the second control word instructions with the length of message and end.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 —	Reserved
14 END	End of Message Indicates the end of DDR message. MSTATUS[COMPLETE] becomes 1 when done. The end can happen before LEN bytes are read if the target ends sooner. If this field is 0, DDR message ends waiting for a new DDR message (issues an HDR restart for the new message). If this field is 1, DDR message ends on HDR exit. 0b - Not the end 1b - End
13-10 —	Reserved
9-0 LEN	Length of Message Contains the length of the message (including the command) in halfwords, up to 2046 bytes. If LEN = 0, then only END is applied: • For reads, + 1 for the CRC. For example, to read 4 bytes (2 halfwords), use 1 + 2 + 1 for CMD + 2 halfwords + CRC. • For writes, LEN is the number of halves of data bytes + 1 (for command).

72.7.49 Controller Write Message Data in DDR mode (MWMSG_DDR_DATA)

Offset

Register	Offset
MWMSG_DDR_DATA	D8h

Function

Contains the 16-bit word to be written in DDR mode. This register functions in a way similar to Controller Write Message in DDR mode: First Control Word (MWMSG_DDR_CONTROL).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA16B															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-0 DATA16B	Data

72.7.50 Controller Read Message in DDR mode (MRMSG_DDR)

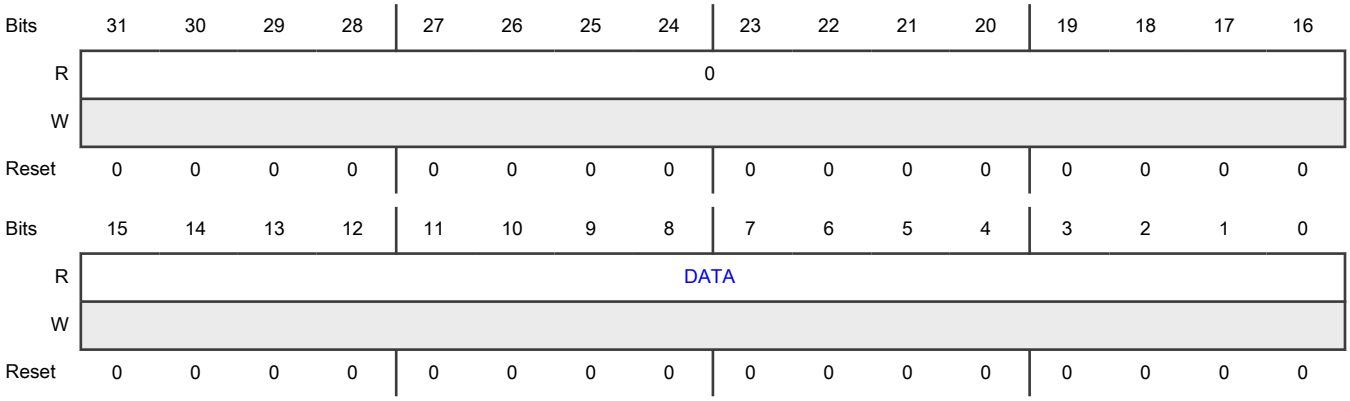
Offset

Register	Offset
MRMSG_DDR	DCh

Function

Allows reading 16-bit words from a target in DDR Message mode from an active message started with MWMSG_DDR. These words are intended to be read by DMA.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 DATA	Data Contains the 16-bit word read from a target. The first byte is the LSB, and is in DATA[7:0]. The second byte is the MSB, and is in DATA[15:8]: • If the target ends before the entire length of the message (MWMSG_DDR[LEN]) is read, the module considers the DATA read as completed. In I3C mode, the target can indicate the end of message (the last byte). Otherwise, the controller terminates the message if the message is more than the controller can accept. • If the target has not yet finished sending DATA before the entire length of the message (MWMSG_DDR[LEN]) is read and END is not a continuation, the DATA read is terminated.

72.7.51 Controller Dynamic Address (MDYNADDR)

Offset

Register	Offset
MDYNADDR	E4h

Function

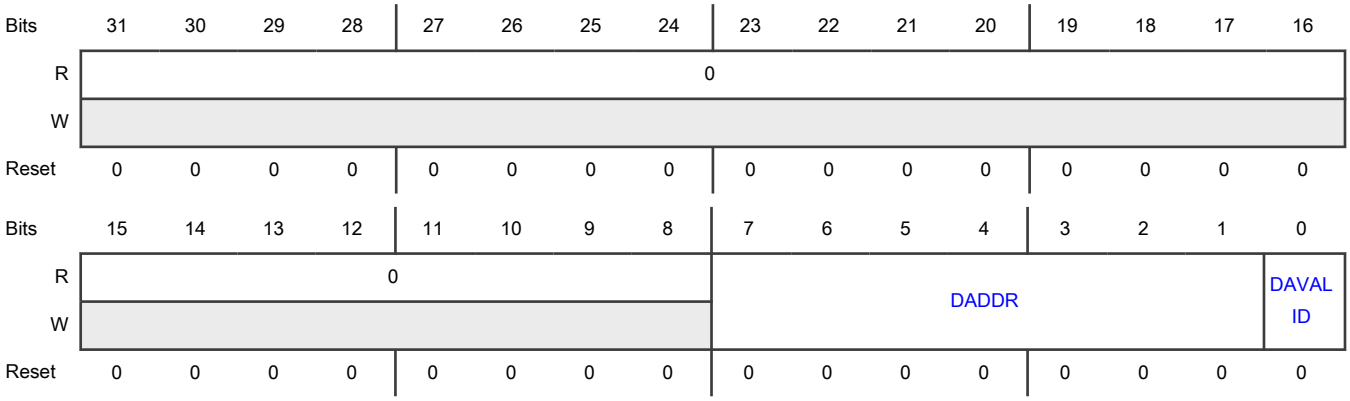
Allows the I3C module to write its own dynamic address (DA) when the module changes from Controller mode to Target mode.

If this device is the main controller (the controller during bus initialization), then the device may use this mechanism to assign itself its DA. When the device hands off control to a secondary controller, it becomes a target itself. This DA must be written before switching to Target mode and must not be changed once in Target mode (it is not clock-safe to do so). It must be written with a valid address value in DADDR if DAVALID = 1.

NOTE

The main controller also uses DEFSLVS CCC to define the target addresses, including itself. This mechanism is how secondary controllers know this address. If the controller is not the main controller, then this mechanism must not be used.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-1 DADDR	Dynamic Address Contains the assigned dynamic address when DAVALID = 1.
0 DAVALID	Dynamic Address Valid 0b - No valid DA assigned 1b - Valid DA assigned

72.7.52 Map Feature Control 0 (SMAPCTRL0)

Offset

Register	Offset
SMAPCTRL0	11Ch

Function

Provides map feature control.

The SMAPCTRL*n* registers are named SMAPCTRL0, SMAPCTRL1, and so on based on the number of mapped addresses. MAPCTRL0 represents the primary DA or SA with SMAPCTRL1 onwards being the mapped addresses.

The features of the SMAPCTRL*n* registers depend on configuration. SMAPCTRL0 acts differently, as described in this chapter.

In general, this mechanism is intended to replace the DYNADDR register for all MAP-related uses.

When using the Auto-MAP and DASA or AASA, the slot is changed from SA to DA. If the controller then issues RSTDAA, the application must rewrite the static addresses and enable them because they are marked disabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				CAUSE				DA							ENA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-11 —	Reserved
10-8 CAUSE	Cause Indicates the cause of the most recent DA assignment, which lead to SSTATUS[DACHG] interrupt. This field has MAP enabled. If this field is 100b (auto MAP change happened last), the change may have changed this DA as well (for example, ENTDAA and SETAASA), but at least one MAP entry automatically changed after that. 000b - No information (this value occurs when not configured to write DA) 001b - Set using ENTDAA 010b - Set using SETDASA, SETAASA, or SETNEWDA 011b - Cleared using RSTDAA 100b - Auto MAP change happened last All other values are reserved.
7-1 DA	Dynamic Address Contains primary DA when ENA = 1. When ENA = 0, static address is used (but not described here). This field has MAP enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 ENA	Enable Primary Dynamic Address Indicates whether the MAP is enabled. 0b - Disabled 1b - Enabled

72.7.53 Extended IBI Data 1 (IBIEXT1)

Offset

Register	Offset
IBIEXT1	140h

Function

Contains extended IBI data.

The CTRL register is used to submit IBI, CR, and Hot-Join when enabled to do so. If allowed, an IBI may have additional bytes following the mandatory data byte (MDB). Extended IBI data is allowed when SCTRL[EXTDATA] = 1. If allowed, the extra bytes are indicated using these registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EXT3								EXT2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EXT1								0	MAX			0	CNT		
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0

Fields

Field	Function
31-24 EXT3	Extra Byte 3 Contains the third extra byte.
23-16 EXT2	Extra Byte 2 Contains the second extra byte.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-8 EXT1	Extra Byte 1 Contains the first extra byte.
7 —	Reserved
6-4 MAX	Maximum Indicates the maximum number of extra bytes allowed by configuration. This field is 0 if there are no extra bytes.
3 —	Reserved
2-0 CNT	Count Contains the number of extra bytes beyond the MDB to be used. This field is 0 if there are no extra bytes.

72.7.54 Extended IBI Data 2 (IBIEXT2)

Offset

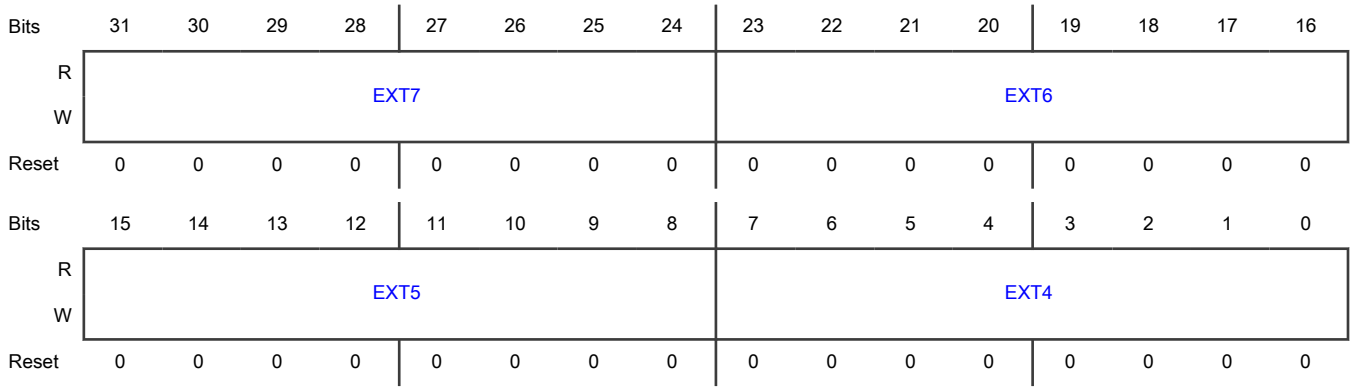
Register	Offset
IBIEXT2	144h

Function

Contains extended IBI data.

The CTRL register is used to submit IBI, CR, and Hot-Join when enabled to do so. If allowed, an IBI may have additional bytes following the mandatory data byte (MDB). Extended IBI data is allowed when SCTRL[EXTDATA] = 1. If allowed, the extra bytes are indicated using these registers.

Diagram



Fields

Field	Function
31-24 EXT7	Extra Byte 7 Contains the seventh extra byte.
23-16 EXT6	Extra Byte 6 Contains the sixth extra byte.
15-8 EXT5	Extra Byte 5 Contains the fifth extra byte.
7-0 EXT4	Extra Byte 4 Contains the forth extra byte.

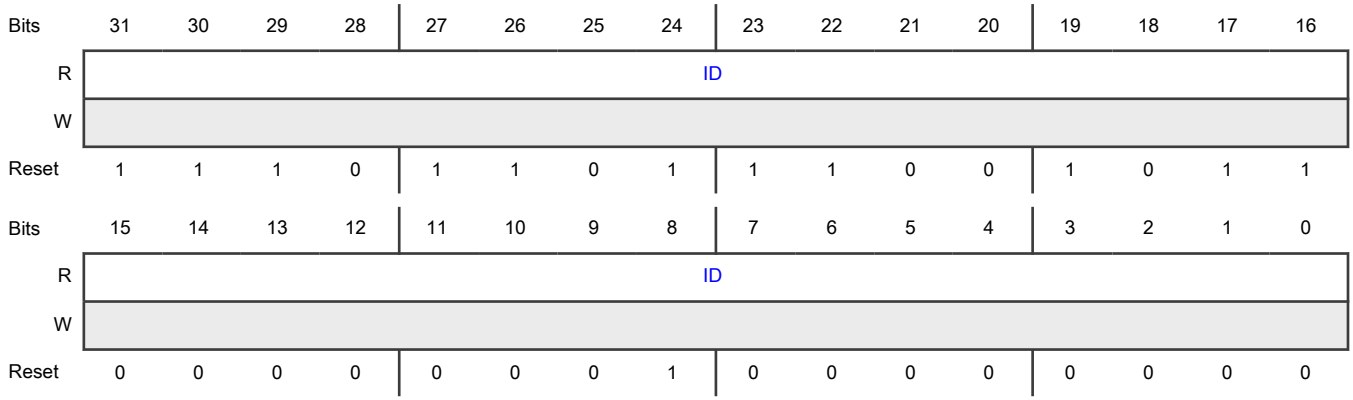
72.7.55 Target Module ID (SID)

Offset

Register	Offset
SID	FFCh

Function
Allows software to detect the module and its version information, if BlockID is enabled.

Diagram



Fields

Field	Function
31-0 ID	ID Indicates the ID. The ID meaning is specific to each use of the I3C module. ID = EDCB0100h.

Chapter 73
General Purpose Input/Output (GPIO)

73.1 Chip-specific GPIO information

Table 704. Reference links to related information

Topic	Related module	Reference
Full description	GPIO	GPIO
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

See Peripheral Reset Control 0 (PRESETCTRL0) to know how to reset GPIO. To enable GPIO clock, see AHB Clock Control 0 (AHBCLKCTRL0).

73.1.1 Module instances

This device has six instances of the GPIO module, GPIO0, GPIO1, GPIO2, GPIO3, GPIO4, and GPIO5.

73.1.2 Security considerations

The GPIO module implements Access protection that can be used to configure secure/non-secure and privileged/non-privileged access on a per pin basis. The interrupt, DMA, or trigger functionality for each pin can also be configured for secure/non-secure and privileged/non-privileged access.

Each GPIO module is also instantiated to use two module slots (for example, GPIO0 and GPIO0_alias). At the Secure AHB controller, one slot can be configured for secure access while the other is configured for non-secure access. This allows each GPIO module to be accessed by both secure and non-secure masters using the appropriate slot, while the Secure AHB controller's MISC_CTRL_REG[DISABLE_STRICT] option is configured for strict mode. Each GPIO module's pin and interrupt, DMA, or trigger secure/non-secure access protections are then used to restrict access according to the level used by that particular GPIO slot.

73.1.3 Interrupt, DMA request, and trigger outputs

GPIO modules can be used to trigger interrupts, DMA requests, or trigger outputs (see the INPUTMUX and WUU chapters for details). See the table below for details on the chip-level interrupt, DMA request, and trigger output capability for each GPIO instantiation:

Table 705. Interrupt, DMA request, and trigger outputs for GPIO

GPIO module	Interrupt	DMA request	Trigger output
GPIO0/GPIO0_alias1	Yes	Yes	No
GPIO1/GPIO1_alias1	Yes	Yes	Yes
GPIO2/GPIO2_alias1	Yes	Yes	Yes
GPIO3/GPIO3_alias1	Yes	Yes	Yes
GPIO4/GPIO4_alias1	Yes	Yes	No
GPIO5/GPIO5_alias1	Yes	Yes	Yes

73.1.4 Power domains/modes

The GPIO module can be implemented in different power domains which has implications for low power mode usage and wake-up capability. See Table 333 and Table 335 in Power Management chapter for GPIO0/1/2/3/4/5 implementation in different power modes.

73.1.5 GPIO wakeup via WAKEUP_b pin

The GPIO module can assert two interrupts into the VBAT module's IRQ0_DET and IRQ1_DET. These interrupts can optionally be enabled as source to assert the WAKEUP_b pin. When the device is operating in the VBAT power mode, you can transition the device back to active mode by configuring an external circuitry to recognize the WAKEUP_b assertion and request for a system power on. See the VBAT chapter for more details.

73.2 Overview

GPIO communicates to the processor core via a zero wait-state interface for maximum pin performance.

73.2.1 Block diagram

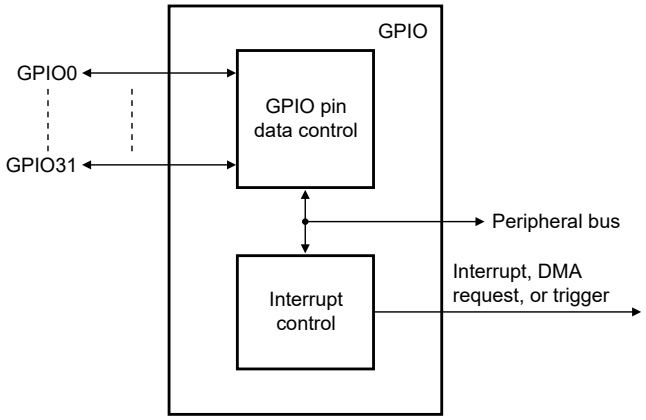


Figure 437. Block diagram

73.2.2 Features

- Port Data Input (PDIR) displays the logic value on each pin when the pin is configured for any digital function, provided the corresponding Port Control and Interrupt modules for that pin are enabled.
- Port Data Output (PDOR), with corresponding set, clear, and toggle registers, controls the output data of each pin when the pin is configured for the GPIO function.
- Port Data Direction (PDDR) controls the direction of each pin when the pin is configured for the GPIO function.
- Port Input Disable (PIDR) controls disabling of the input for each general-purpose pin.
- Pin interrupts:
  - Interrupt flags and enable registers for each pin are functional in all digital pin muxing modes.
  - Support for interrupt, peripheral trigger, or DMA request is configured for each pin.
  - Support for edge-sensitive (rising, falling, or both) or level-sensitive (low, high) interrupts is configured for each pin.
  - Asynchronous wake-up in Low-Power mode.
  - GPIO generates a total of 2 interrupts, 2 output triggers, and 2 DMA requests.
  - Each pin can be used to generate a single interrupt, output trigger, or DMA request.

- Protection registers:
  - Each pin is configured for secure or nonsecure and privileged or nonprivileged access.
  - Each interrupt, trigger, and DMA request domain is configured for secure or nonsecure and privileged or nonprivileged access.

73.3 Functional description

73.3.1 Low-Power mode

You can configure GPIO to exit Low-Power mode via an asynchronous wake-up signal if an enabled interrupt is detected.

73.3.2 Debug mode

GPIO remains functional in Debug mode.

73.3.3 General-purpose input

The logic state of each pin is available via Port Data Input (PDIR) if:

- The corresponding field in Port Input Disable (PIDR) is 0.
- The pin is configured for a digital function.

73.3.4 General-purpose output

The logic state of each pin is controlled via Port Data Output (PDOR) and Port Data Direction (PDDR), provided the pin is configured for the GPIO function. The following table depicts the conditions for a pin to be configured as input or output.

Table 706. General-purpose output

If	Then
A pin is configured for the GPIO function and the corresponding PDDR field is 0	The pin is configured as an input.
A pin is configured for the GPIO function and the corresponding PDDR field is 1	The pin is configured as an output and the logic state of the pin is equal to the corresponding PDOR field.

For efficient bit manipulation on the general-purpose outputs, pin data set, pin data clear, and pin data toggle registers allow one or more outputs within one port to be set, cleared, or toggled from a single register write, eliminating the need for a read-modify-write operation to prevent changing pins accidentally.

73.3.5 Clocking

GPIO receives a single clock, which is used for register access and synchronization with external pin inputs. There are no special considerations.

73.3.6 Reset

GPIO receives a single reset, which resets the peripheral. There are no special considerations.

73.3.7 External interrupts

The external interrupt capability of GPIO is available in all digital pin muxing modes.

GPIO generates a total of 2 interrupts, 2 output triggers, and 2 DMA requests. You can configure each pin individually for interrupt, output trigger, or DMA requests:

- Each output implements a separate interrupt status flag (ISF) register for that domain.

- Each output generates a single interrupt.
- Each output generates a single DMA request.
- Each output generates a single peripheral trigger output.

You can configure each pin individually for any of the external interrupt modes shown in the following table.

Table 707. Available pin configurations for external interrupts

Signal conditions	Software polling using flags	Peripheral triggers	Interrupts	DMA requests
Rising-edge	Yes	—	Yes	Yes
Falling-edge	Yes	—	Yes	Yes
Rising- and falling-edge	Yes	—	Yes	Yes
High-level	—	Yes	Yes	—
Low-level	—	Yes	Yes	—

The interrupt status flag is set when the configured edge or level is detected on the pin. Unless GPIO is in Low-Power mode, the input is first synchronized to the system clock to detect the configured level or edge transition.

GPIO generates a pin interrupt that asserts when the interrupt status flag is set for any enabled interrupt for that output. The interrupt negates after the interrupt status flags for all enabled interrupts are cleared by writing a logic 1 to either ISFR0[ISF*n*] or ICR0[ISF].

GPIO generates a DMA request that asserts when the interrupt status flag is set for any enabled DMA request in that output. The DMA request negates after the DMA transfer is completed because that clears the interrupt status flags for all enabled DMA requests.

In Low-Power mode, the interrupt status flag for any enabled interrupt is asynchronously set if the required level or edge is detected. This also generates an asynchronous wake-up signal to exit Low-Power mode.

GPIO generates a peripheral trigger output that asserts if any pin configured for the active-high trigger is logic 1, or any pin triggered for the active-low trigger is logic 0. The peripheral trigger output asynchronously updates from the value on the configured pins.

73.3.8 Global interrupt control

The two global interrupt control registers (Global Interrupt Control Low (GICLR) and Global Interrupt Control High (GICHR)) allow a single register write to update the upper 16 bits of Interrupt Control a (ICR0 - ICR31) for up to 16 pins, all with the same value.

These global interrupt control registers allow you to quickly configure multiple pins within the same port with the same interrupt configuration.

The global interrupt control registers are write-only registers and always read 0.

73.3.9 DMA

GPIO generates 2 requests. For each pin, you can configure DMA requests to assert for either rising, falling, or both edge detection. See External interrupts for more information.

73.3.10 Access protection

GPIO implements access protection registers (PCNS and PCNP) for each pin, as shown in Table 708.

Table 708. PCNS and PCNP register access protection implementation

Access	Description
Secure	You can write to or read the pins configured for secure access only in the Secure state.
Nonsecure	You can write to or read the pins configured for nonsecure access only in the Nonsecure state.
Privilege	You can write to the pins configured for privilege access only in the Privilege state.
Nonprivilege	You can write to the pins configured for nonprivilege access in both the Privilege and Nonprivilege states.

GPIO implements access protection registers (ICNS and ICNP) for each interrupt, trigger output, and DMA request as shown in Table 709.

Table 709. ICNS and ICNP register access protection implementation

Access	Description
Secure	You can configure the outputs programmed for secure access only in the Secure state.
Nonsecure	You can configure the outputs programmed for nonsecure access only in the Nonsecure state.
Privilege	You can configure the outputs programmed for privilege access only in the Privilege state.
Nonprivilege	You can configure the outputs programmed for nonprivilege access in both the Privilege and Nonprivilege states.

You can write to the access protection registers (PCNS, PCNP, ICNS, and ICNP) only in the Secure-Privilege state; you can optionally lock these registers until the next reset.

Configuring a pin interrupt, trigger output, or DMA request requires the following access permissions:

- Access permission to configure the pin.
- Access permission to either configure to the desired interrupt, trigger output, or DMA request, or writing 0 to ICR*n*[IRQC].
- Access permission to either configure with the existing interrupt, trigger output, or DMA request, or if the current value of ICR*n*[IRQC] is 0.
- The interrupt configuration is not locked.

73.4 External signals

Table 710. External signals

Signal	Description		Direction
GPIO31–GPIO0	General-purpose input/output		I/O
	State meaning	Asserted: The pin is logic 1. Deasserted: The pin is logic 0.	
	Timing	Assertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock. Deassertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock.	

NOTE

Not all pins within each GPIO are implemented on each chip. See the "Signal Multiplexing" chapter for the number of GPIO pins within each port available on this chip.

73.5 Initialization

To initialize GPIO:

1. Initialize the GPIO pins for the output function:
   a. Configure the output logic value for each pin by using Port Data Output (PDOR).
   b. Configure the direction for each pin by using Port Data Direction (PDDR).
2. Initialize the interrupt function by writing to Interrupt Control a (ICR0 - ICR31) for the corresponding pins and desired configuration. If the pin is previously used for a different function, first write 0100_0000h to Interrupt Control a (ICR0 - ICR31) to disable the previous function and clear the flag.

73.6 Application information

GPIO includes the following applications:

- Reading the state of a single pin by performing a byte read of Pin Data (P0DR - P31DR).
- Updating the state of a single pin by performing a byte write to Pin Data (P0DR - P31DR).
- Reading the state of multiple pins by reading Port Data Input (PDIR).
- Updating the state of multiple pins by using the following ways:
  - Writing to Port Data Output (PDOR).
  - Writing to Port Set Output (PSOR) to write 1 to Port Data Output (PDOR).
  - Writing to Port Clear Output (PCOR) to write 0 to Port Data Output (PDOR).
  - Writing to Port Toggle Output (PTOR) to toggle Port Data Output (PDOR).

73.7 Memory map and register definition

The GPIO registers support 8-bit, 16-bit, or 32-bit accesses. Any read or write access to the GPIO memory space that is outside the valid memory map results in a bus error.

NOTE

For simplicity, each GPIO port's register appears with the same width of 32 bits, corresponding to 32 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is chip-specific. See the "Signal Multiplexing" chapter for the exact control bits of each port.

73.7.1 GPIO register descriptions

73.7.1.1 GPIO memory map

GPIO0 base address: 4009_6000h

GPIO0_alias1 base address: 4009_7000h

GPIO1 base address: 4009_8000h

GPIO1_alias1 base address: 4009_9000h

GPIO2 base address: 4009_A000h

GPIO2_alias1 base address: 4009_B000h

GPIO3 base address: 4009_C000h

GPIO3_alias1 base address: 4009_D000h

GPIO4 base address: 4009_E000h

GPIO4_alias1 base address: 4009_F000h

GPIO5 base address: 4004_0000h

GPIO5_alias1 base address: 4004_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0201_0001h
4h	Parameter (PARAM)	32	R	0000_0002h
Ch	Lock (LOCK)	32	RW	0000_0000h
10h	Pin Control Nonsecure (PCNS)	32	RW	0000_0000h
14h	Interrupt Control Nonsecure (ICNS)	32	RW	0000_0000h
18h	Pin Control Nonprivilege (PCNP)	32	RW	0000_0000h
1Ch	Interrupt Control Nonprivilege (ICNP)	32	RW	0000_0000h
40h	Port Data Output (PDOR)	32	RW	0000_0000h
44h	Port Set Output (PSOR)	32	RW	0000_0000h
48h	Port Clear Output (PCOR)	32	RW	0000_0000h
4Ch	Port Toggle Output (PTOR)	32	RW	0000_0000h
50h	Port Data Input (PDIR)	32	R	0000_0000h
54h	Port Data Direction (PDDR)	32	RW	0000_0000h
58h	Port Input Disable (PIDR)	32	RW	0000_0000h
60h - 7Fh	Pin Data (P0DR - P31DR)	8	RW	00h
80h - FCh	Interrupt Control a (ICR0 - ICR31)	32	RW	0000_0000h
100h	Global Interrupt Control Low (GICLR)	32	RW	0000_0000h
104h	Global Interrupt Control High (GICHR)	32	RW	0000_0000h
120h - 124h	Interrupt Status Flag (ISFR0 - ISFR1)	32	RW	0000_0000h

73.7.1.2 Version ID (VERID)

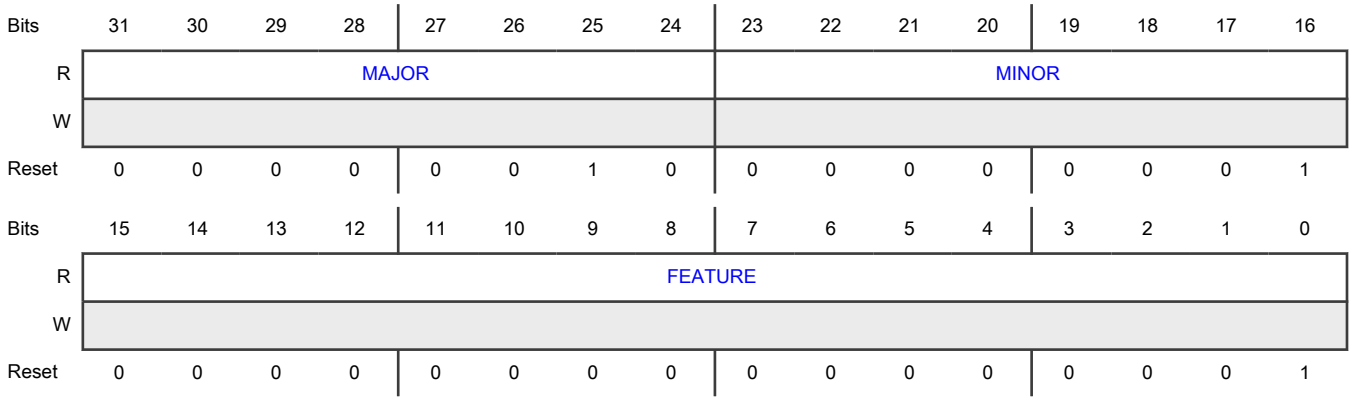
Offset

Register	Offset
VERID	0h

Function

Indicates the version ID number of each GPIO.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Indicates the major version number for the specification.
23-16 MINOR	Minor Version Number Indicates the minor version number for the specification.
15-0 FEATURE	Feature Specification Number Indicates the feature set number. 0000_0000_0000_0000b - Basic implementation 0000_0000_0000_0001b - Protection registers implemented

73.7.1.3 Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

Function

Indicates the interrupt number of each GPIO.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												IRQNUM			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-4 —	Reserved
3-0 IRQNUM	Interrupt Number Indicates the number of interrupt, trigger, or DMA request domains.

73.7.1.4 Lock (LOCK)

Offset

Register	Offset
LOCK	Ch

Function

Locks the nonsecure and nonprivilege access protection registers. You can write to this register only in the Secure-Privilege state.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												ICNP	PCNP	ICNS	PCNS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-4 —	Reserved
3 ICNP	Lock ICNP Locks Interrupt Control Nonprivilege (ICNP). If this field is 0, you can write to ICNP in the Secure-Privilege state. If this field is 1, you cannot write to ICNP until the next reset. 0b - Writable in Secure-Privilege state 1b - Not writable until the next reset
2 PCNP	Lock PCNP Locks Pin Control Nonprivilege (PCNP). If this field is 0, you can write to PCNP in the Secure-Privilege state. If this field is 1, you cannot write to PCNP until the next reset. 0b - Writable in Secure-Privilege state 1b - Not writable until the next reset
1 ICNS	Lock ICNS Locks Interrupt Control Nonsecure (ICNS). If this field is 0, you can write to ICNS in the Secure-Privilege state. If this field is 1, you cannot write to ICNS until the next reset. 0b - Writable in Secure-Privilege state 1b - Not writable until the next reset
0 PCNS	Lock PCNS Locks Pin Control Nonsecure (PCNS). If this field is 0, you can write to PCNS in the Secure-Privilege state. If this field is 1, you cannot write to PCNS until the next reset. 0b - Writable in Secure-Privilege state 1b - Not writable until the next reset

73.7.1.5 Pin Control Nonsecure (PCNS)

Offset

Register	Offset
PCNS	10h

Function

Configures secure or nonsecure access protection for each pin. You can write to this register only in the Secure-Privilege state if it is not locked (LOCK[PCNS] = 0).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NSE31	NSE30	NSE29	NSE28	NSE27	NSE26	NSE25	NSE24	NSE23	NSE22	NSE21	NSE20	NSE19	NSE18	NSE17	NSE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NSE15	NSE14	NSE13	NSE12	NSE11	NSE10	NSE9	NSE8	NSE7	NSE6	NSE5	NSE4	NSE3	NSE2	NSE1	NSE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 NSEn	Nonsecure Enable Configures secure or nonsecure access protection for each pin. If this field is 0, the pin is configured for secure access. You can read or write to the corresponding pin's registers and fields only in the Secure state. When you access the corresponding pin's registers in the Nonsecure state, all fields in the registers related to that pin are read zero (with writes ignored). If this field is 1, the pin is configured for nonsecure access. You can read or write to the corresponding pin's registers and fields only in the Nonsecure state. When you access the corresponding pin's registers in the Secure state, all fields in the registers related to that pin are read zero (with writes ignored). 0b - Secure access 1b - Nonsecure access

73.7.1.6 Interrupt Control Nonsecure (ICNS)

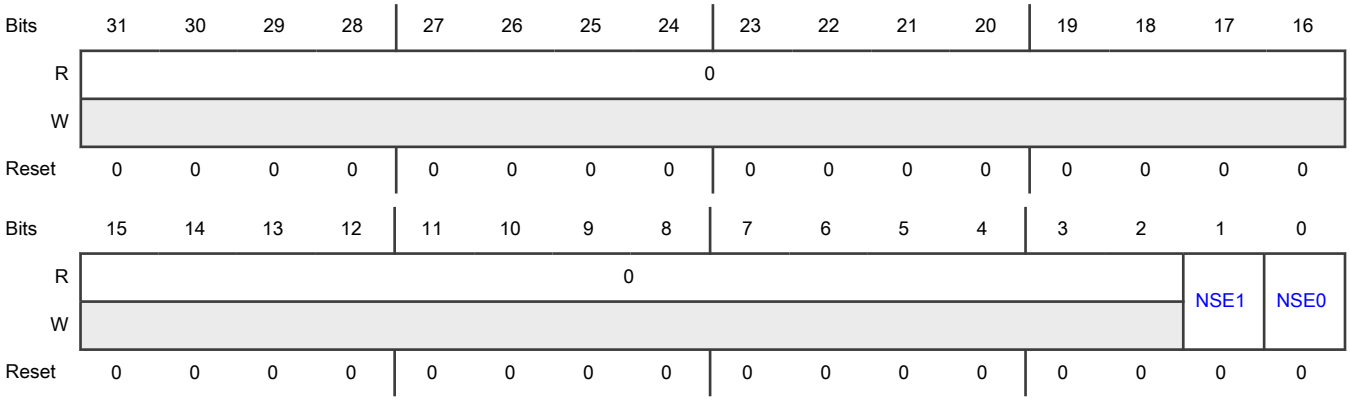
Offset

Register	Offset
ICNS	14h

Function

Configures secure and nonsecure access protection for each interrupt, output trigger, or DMA request. You can update this register only in the Secure-Privilege state if it is not locked (LOCK[ICNS] = 0).

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 NSEn	Nonsecure Enable Configures secure or nonsecure access protection for each interrupt, output trigger, or DMA request. If this field is 0, the interrupt, output trigger, or DMA request is configured for secure access. You can configure a pin to use the corresponding interrupt, output trigger, or DMA request, or reconfigure a pin that is already configured to use the corresponding interrupt, output trigger, or DMA request only in the Secure state. If this field is 1, the interrupt, output trigger, or DMA request is configured for nonsecure access. You can configure a pin to use the corresponding interrupt, output trigger, or DMA request, or reconfigure a pin that is already configured to use the corresponding interrupt, output trigger, or DMA request only in the Nonsecure state. **NOTE** See the GPIO chip-specific information to determine which GPIO instances support interrupt, DMA request, or trigger capabilities. 0b - Secure access 1b - Nonsecure access

73.7.1.7 Pin Control Nonprivilege (PCNP)

Offset

Register	Offset
PCNP	18h

Function

Configures access protection on each pin either for privilege or for both privilege and nonprivilege accesses. You can update this register only in the Secure-Privilege state if it is not locked (LOCK[PCNP] = 0).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NPE31	NPE30	NPE29	NPE28	NPE27	NPE26	NPE25	NPE24	NPE23	NPE22	NPE21	NPE20	NPE19	NPE18	NPE17	NPE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NPE15	NPE14	NPE13	NPE12	NPE11	NPE10	NPE9	NPE8	NPE7	NPE6	NPE5	NPE4	NPE3	NPE2	NPE1	NPE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 NPEn	Nonprivilege Enable Configures privilege or nonprivilege access protection for each pin. If this field is 0, the pin is configured for privilege access. Write access to the corresponding pin's registers and fields is allowed only in the Privilege state. When you access the corresponding pin's registers and fields in the Nonprivilege state, all fields related to that pin in this GPIO are readable (with writes ignored). If this field is 1, the pin is configured for nonprivilege access; read or write access to the corresponding pin's registers is allowed in both Privilege and Nonprivilege states. 0b - Privilege access 1b - Nonprivilege access

73.7.1.8 Interrupt Control Nonprivilege (ICNP)

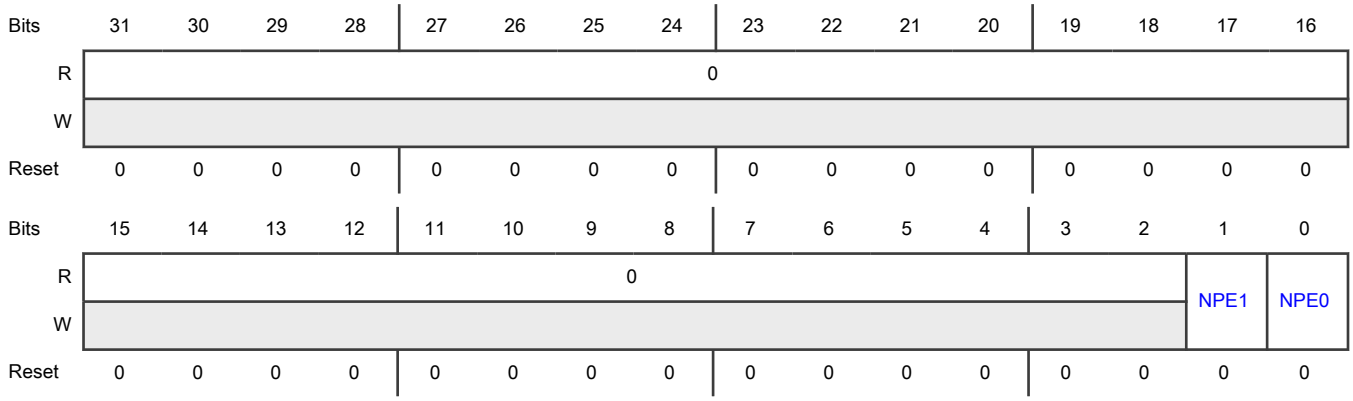
Offset

Register	Offset
ICNP	1Ch

Function

Configures privilege and nonprivilege access protection for each interrupt, trigger output, or DMA request. You can update this register only in the Secure-Privilege state if it is not locked (LOCK[ICNP] = 0).

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 NPEn	Nonprivilege Enable Configures privilege or nonprivilege access protection for each interrupt, trigger output, or DMA request. If this field is 0, the pin is configured for privilege access. You can configure a pin to use the corresponding interrupt, trigger output, or DMA request, or reconfigure a pin that is already configured to use the corresponding interrupt, trigger output, or DMA request only in the Privilege state. If this field is 1, the pin is configured for nonprivilege access. In either Privilege or Nonprivilege state, you can configure a pin to use the corresponding interrupt, trigger output, or DMA request, or reconfigure a pin that is already configured to use the corresponding interrupt, trigger output, or DMA request. **NOTE** See the GPIO chip-specific information to determine which GPIO instances support interrupt, DMA request, or trigger capabilities. 0b - Privilege access 1b - Nonprivilege access

73.7.1.9 Port Data Output (PDOR)

Offset

Register	Offset
PDOR	40h

Function

Configures the logic levels that are driven on each general-purpose output pin.

NOTE

Do not modify the pin configuration registers associated with pins that are not available in your selected package. By default, these unbonded pins are set to the Disable state for lowest power consumption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PDO31	PDO30	PDO29	PDO28	PDO27	PDO26	PDO25	PDO24	PDO23	PDO22	PDO21	PDO20	PDO19	PDO18	PDO17	PDO16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDO15	PDO14	PDO13	PDO12	PDO11	PDO10	PDO9	PDO8	PDO7	PDO6	PDO5	PDO4	PDO3	PDO2	PDO1	PDO0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PDOn	Port Data Output Configures the logic level on the pin if it is configured for general-purpose output. If this field is 0, logic level 0 is driven on the pin, if the pin is configured for general-purpose output. If this field is 1, logic level 1 is driven on the pin, if the pin is configured for general-purpose output. **NOTE** Reading the fields for unbonded pins returns an undefined value. 0b - Logic level 0 1b - Logic level 1

73.7.1.10 Port Set Output (PSOR)

Offset

Register	Offset
PSOR	44h

Function

Updates the corresponding fields of Port Data Output (PDOR) to become 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PTSO 31	PTSO 30	PTSO 29	PTSO 28	PTSO 27	PTSO 26	PTSO 25	PTSO 24	PTSO 23	PTSO 22	PTSO 21	PTSO 20	PTSO 19	PTSO 18	PTSO 17	PTSO 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PTSO 15	PTSO 14	PTSO 13	PTSO 12	PTSO 11	PTSO 10	PTSO 9	PTSO 8	PTSO 7	PTSO 6	PTSO 5	PTSO 4	PTSO 3	PTSO 2	PTSO 1	PTSO 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PTSOn	Port Set Output Updates the content of the corresponding field in Port Data Output (PDOR). If this field is 0, the corresponding PDOR field does not change. If this field is 1, the corresponding PDOR field becomes 1. 0b - No change 1b - Corresponding field in PDOR becomes 1

73.7.1.11 Port Clear Output (PCOR)

Offset

Register	Offset
PCOR	48h

Function

Updates the corresponding fields of Port Data Output (PDOR) to become 0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PTCO 31	PTCO 30	PTCO 29	PTCO 28	PTCO 27	PTCO 26	PTCO 25	PTCO 24	PTCO 23	PTCO 22	PTCO 21	PTCO 20	PTCO 19	PTCO 18	PTCO 17	PTCO 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PTCO 15	PTCO 14	PTCO 13	PTCO 12	PTCO 11	PTCO 10	PTCO 9	PTCO 8	PTCO 7	PTCO 6	PTCO 5	PTCO 4	PTCO 3	PTCO 2	PTCO 1	PTCO 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PTCOn	Port Clear Output Updates the content of the corresponding field in Port Data Output (PDOR). If this field is 0, the corresponding PDOR field does not change. If this field is 1, the corresponding PDOR field becomes 0. 0b - No change 1b - Corresponding field in PDOR becomes 0

73.7.1.12 Port Toggle Output (PTOR)

Offset

Register	Offset
PTOR	4Ch

Function

Updates the corresponding fields of Port Data Output (PDOR) to set to the inverse of their current logic states.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PTTO 31	PTTO 30	PTTO 29	PTTO 28	PTTO 27	PTTO 26	PTTO 25	PTTO 24	PTTO 23	PTTO 22	PTTO 21	PTTO 20	PTTO 19	PTTO 18	PTTO 17	PTTO 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PTTO 15	PTTO 14	PTTO 13	PTTO 12	PTTO 11	PTTO 10	PTTO 9	PTTO 8	PTTO 7	PTTO 6	PTTO 5	PTTO 4	PTTO 3	PTTO 2	PTTO 1	PTTO 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PTTOn	Port Toggle Output Updates the content of the corresponding field in Port Data Output (PDOR). If this field is 0, the corresponding PDOR field does not change. If this field is 1, the corresponding PDOR field is set to the inverse of its current logic state. 0b - No change 1b - Set to the inverse of its current logic state

73.7.1.13 Port Data Input (PDIR)

Offset

Register	Offset
PDIR	50h

Function

Captures the logic levels of each general-purpose input pin.

NOTE

Do not modify the pin configuration registers associated with the pins that are not available in your selected package. By default, these unbonded pins are set to the Disable state for lowest power consumption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PDI31	PDI30	PDI29	PDI28	PDI27	PDI26	PDI25	PDI24	PDI23	PDI22	PDI21	PDI20	PDI19	PDI18	PDI17	PDI16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PDI15	PDI14	PDI13	PDI12	PDI11	PDI10	PDI9	PDI8	PDI7	PDI6	PDI5	PDI4	PDI3	PDI2	PDI1	PDI0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PDIn	Port Data Input Indicates the logic level of the pin that is configured for use by a digital function. If this field is 0, the pin logic level is logic 0 or is not configured or implemented for use by a digital function. If this field is 1, the pin logic level is logic 1. 0b - Logic 0 1b - Logic 1

73.7.1.14 Port Data Direction (PDDR)

Offset

Register	Offset
PDDR	54h

Function

Configures the individual port pins for input or output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PDD31	PDD30	PDD29	PDD28	PDD27	PDD26	PDD25	PDD24	PDD23	PDD22	PDD21	PDD20	PDD19	PDD18	PDD17	PDD16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDD15	PDD14	PDD13	PDD12	PDD11	PDD10	PDD9	PDD8	PDD7	PDD6	PDD5	PDD4	PDD3	PDD2	PDD1	PDD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PDDn	Port Data Direction Configures individual port pins for input or output. If this field is 0, the pin is configured as general-purpose input for the GPIO function. If this field is 1, the pin is configured as general-purpose output for the GPIO function. 0b - Input 1b - Output

73.7.1.15 Port Input Disable (PIDR)

Offset

Register	Offset
PIDR	58h

Function

Disables the input for each general-purpose pin, which prevents the value from being reported in Port Data Input (PDIR).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24	PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8	PID7	PID6	PID5	PID4	PID3	PID2	PID1	PID0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 PIDn	Port Input Disable Disables a pin for general-purpose input. If this field is 0, the pin is configured for general-purpose input, provided the pin is configured for a digital function. If this field is 1, the pin is disabled for general-purpose input. 0b - Configured for general-purpose input 1b - Disabled for general-purpose input

73.7.1.16 Pin Data (P0DR - P31DR)

Offset

For a = 0 to 31:

Register	Offset
PaDR	60h + (a × 1h)

Function

Configures the data feature of a pin. Pins that are unimplemented or not configured for a digital function read zero.

NOTE

You must not modify Pin Data (P0DR - P31DR) associated with the pins that are not available in your selected package. These unbonded pins are, by default, set to the Disable state for lowest power consumption.

Diagram

Bits	7	6	5	4	3	2	1	0
R	0							PD
W								
Reset	0	0	0	0	0	0	0	0

Fields

Field	Function
7-1 —	Reserved
0 PD	Pin Data (I/O) Specifies the pin logic level. This field updates the corresponding field in Port Data Output (PDOR); reading this field returns the value in the corresponding field of Port Data Input (PDIR). If this field is 0, the pin logic level is logic zero or not configured for use by a digital function. If this field is 1, the pin logic level is logic one. 0b - Logic zero 1b - Logic one

73.7.1.17 Interrupt Control a (ICR0 - ICR31)

Offset

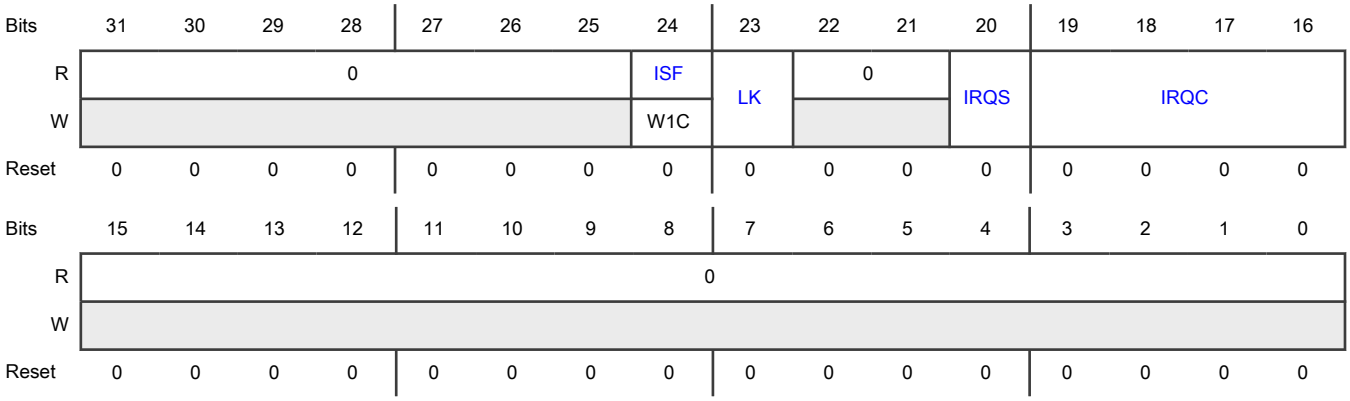
For a = 0 to 31:

Register	Offset
ICRa	80h + (a × 4h)

Function

Configures interrupt features on each pin.

Diagram



Fields

Field	Function
31-25 —	Reserved
24 ISF	Interrupt Status Flag Indicates whether the configured interrupt is detected. The pin interrupt configuration is valid in all digital pin muxing modes. The fields in Interrupt Status Flag (ISFR0 - ISFR1) have the same function. ISF can be cleared with either register field. If the pin is configured to generate a DMA request, then the corresponding flag is cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level-sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag
23 LK	Lock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Locks ICR[23:0], and the locked field cannot be updated until the next system reset. If this field is 0, interrupt configuration by ICR[23:0] is not locked and can be updated. If this field is 1, interrupt configuration by ICR[23:0] is locked and cannot be updated until the next system reset. 0b - Lock 1b - Do not lock
22-21 —	Reserved
20 IRQS	Interrupt Select Configures the selected interrupt, trigger output, or DMA request. **NOTE** See the GPIO chip-specific information to determine which GPIO instances support interrupt, DMA request, or trigger capabilities. 0b - Interrupt, trigger output, or DMA request 0 1b - Interrupt, trigger output, or DMA request 1
19-16 IRQC	Interrupt Configuration Specifies the ISF and DMA request configuration. The pin interrupt configuration is valid in all digital pin muxing modes. When changing the interrupt configuration, it is recommended to first disable ISF and then write the new configuration. The corresponding pin is configured to generate interrupt, trigger, or DMA request. **NOTE** See the GPIO chip-specific information to determine the GPIO instances that support interrupt, DMA request, or trigger capabilities. 0000b - ISF is disabled 0001b - ISF and DMA request on rising edge 0010b - ISF and DMA request on falling edge 0011b - ISF and DMA request on either edge 0100b - Reserved 0101b - ISF sets on rising edge 0110b - ISF sets on falling edge 0111b - ISF sets on either edge 1000b - ISF and interrupt when logic 0 1001b - ISF and interrupt on rising edge 1010b - ISF and interrupt on falling edge 1011b - ISF and Interrupt on either edge

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1100b - ISF and interrupt when logic 1 1101b - Enable active-high trigger output; ISF on rising edge (pin state is ORed with other enabled triggers to generate the output trigger for use by other peripherals) 1110b - Enable active-low trigger output; ISF on falling edge (pin state is inverted and ORed with other enabled triggers to generate the output trigger for use by other peripherals) 1111b - Reserved
15-0 —	Reserved

73.7.1.18 Global Interrupt Control Low (GICLR)

Offset

Register	Offset
GICLR	100h

Function

Updates any combination of the lower 16 Interrupt Control registers with the same value. This register supports only 32-bit writes and ignores any 16-bit or 8-bit writes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	GIWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	GIWE 15	GIWE 14	GIWE 13	GIWE 12	GIWE 11	GIWE 10	GIWE 9	GIWE 8	GIWE 7	GIWE 6	GIWE 5	GIWE 4	GIWE 3	GIWE 2	GIWE 1	GIWE 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 GIWD	Global Interrupt Write Data

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates the write value that is written to the upper 16 bits of Interrupt Control a (ICR0 - ICR31), selected by GIWE.
15-0 GIWEn	Global Interrupt Write Enable Indicates whether the upper 16 bits of the corresponding Interrupt Control a (ICR0 - ICR31) are updated with the value in GIWD. 0b - Not updated 1b - Updated

73.7.1.19 Global Interrupt Control High (GICHR)

Offset

Register	Offset
GICHR	104h

Function

Updates any combination of the upper 16 Interrupt Control registers with the same value. This register supports only 32-bit writes and ignores any 16-bit or 8-bit writes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	GIWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	GIWE 31	GIWE 30	GIWE 29	GIWE 28	GIWE 27	GIWE 26	GIWE 25	GIWE 24	GIWE 23	GIWE 22	GIWE 21	GIWE 20	GIWE 19	GIWE 18	GIWE 17	GIWE 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 GIWD	Global Interrupt Write Data Indicates the write value that is written to the upper 16 bits of Interrupt Control a (ICR0 - ICR31), selected by GIWE.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-0 GIWEn	Global Interrupt Write Enable Indicates whether the upper 16 bits of the corresponding Interrupt Control a (ICR0 - ICR31) are updated with the value in GIWD. 0b - Not updated. 1b - Updated

73.7.1.20 Interrupt Status Flag (ISFR0 - ISFR1)

Offset

Register	Offset
ISFR0	120h
ISFR1	124h

Function

Indicates whether the related configured interrupt is detected on each pin. The pin interrupt configuration is valid in all digital pin muxing modes. The ISF for each pin is also visible in the corresponding Interrupt Control register, and each flag can be cleared in either location.

There is a separate ISF register for each interrupt, trigger, or DMA request domain. Each status flag is only visible in the register that corresponds to that flag's domain, as configured in ICR*n*[IRQS].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ISF31	ISF30	ISF29	ISF28	ISF27	ISF26	ISF25	ISF24	ISF23	ISF22	ISF21	ISF20	ISF19	ISF18	ISF17	ISF16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISF15	ISF14	ISF13	ISF12	ISF11	ISF10	ISF9	ISF8	ISF7	ISF6	ISF5	ISF4	ISF3	ISF2	ISF1	ISF0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 ISFn	Interrupt Status Flag

Table continues on the next page...

Field	Function
	Indicates the detection of the configured interrupt on each pin of the same number. If this field is 0, the configured interrupt is not detected on the pin of the same number. If this field is 1, the configured interrupt is detected on the pin of the same number. If the pin is configured to generate a DMA request, then the corresponding flag is cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level-sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. **NOTE** This field behaves differently for register reads and writes. When reading 0b - Not detected 1b - Detected When writing 0b - No effect 1b - Clear the flag

Chapter 74
Pin Interrupt and Pattern Match (PINT)

74.1 Chip-specific PINT information

Table 711. Reference links to related information

Topic	Related module	Reference
Full description	PINT	PINT
System memory map		Memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
Input multiplexing	INPUTMUX	See PINTSEL0 - PINTSEL7 registers in INPUTMUX

74.1.1 Module instances

This device has one instance of the PINT module, PINT0.

74.1.2 Initialization

1. Select up to eight external interrupt pins from all available digital port pins on ports 0 and 1 in INPUTMUX. The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
2. Enable the clock to the PINT module via SYSCON AHBCLKCTRL0[PINT].
3. Clear the PINT peripheral reset via SYSCON PRESETCTRL0[PINT_RST].

74.1.2.1 Pin interrupts and Pattern match initialization

In addition to the initialization steps described above, for Pin interrupts initialization and Pattern match initialization, ensure that eight interrupt outputs from PINT are ORed together to one NVIC slot.

74.1.2.2 Configure pins as pin interrupts or as inputs to the pattern match engine

1. Determine the pins that you want to serve as pin interrupts or pattern match inputs. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin selected, program the GPIO port pin number from ports 0 and 1 into one of the eight PINT_SEL registers in the INPUTMUX module.

NOTE

The port pin number serves to identify the pin to the PINT_SEL register. Any function, including GPIO, can be assigned to this pin via IOCON.

3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can be set up.

NOTE

The inputs to the Pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

74.2 Overview

The Pin Interrupt and Pattern Match (PINT) IP provides interrupt functionality for a selected set of GPIO pins. Per-pin edge- or level-sensitive interrupts can be generated. This IP also includes a pattern match engine that uses the selected GPIO pins as inputs to a Boolean expression. When the expression is satisfied, an interrupt is generated.

74.2.1 Block diagram

From all available GPIO pins, select up to 8 pins in the INPUTMUX module to serve as external interrupt pins (see the "INPUTMUX" chapter for more details). Each external interrupt pins connect to eight individual interrupts in the NVIC. It is possible to create external interrupt pins based either on rising or falling edges or on the input level.

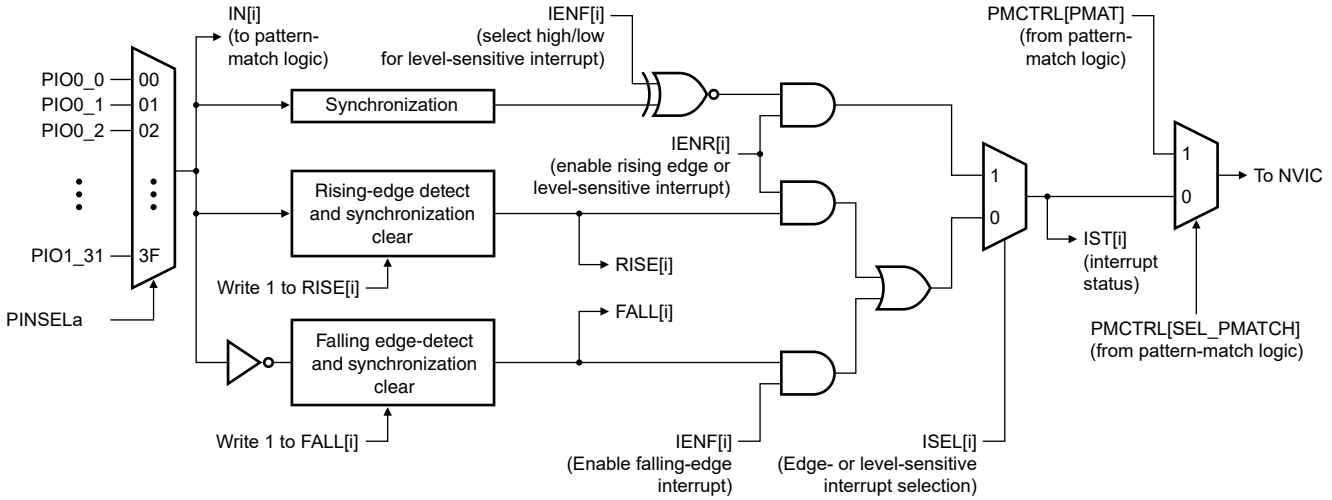


Figure 438. Pin Interrupt diagram

74.2.2 Features

- Pin interrupts
  - Select up to 8 pins from all GPIO pins on ports 0 and 1 as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
  - Edge-sensitive interrupt pins interrupt on rising or falling edges or both.
  - Level-sensitive interrupt pins are high- or low-active.
- Pattern-match engine
  - Select up to 8 pins from all digital pins on ports 0 and 1 to contribute to a Boolean expression. The Boolean expression consists of specified levels and/or transitions on various combinations of these pins.
  - Each bit slice minterm (product term) consisting of the specified Boolean expression can generate its own, dedicated interrupt request.
  - You can program any occurrence of a pattern match to generate a Receive Event (RXEV) notification to the CPU.
  - Use pattern match, in conjunction with software, to create complex state machines based on pin inputs.

74.3 Functional description

74.3.1 Pattern-match engine

With pattern matching, you can construct complex Boolean expressions using the same set of 8 GPIO pins you selected for interrupts. Implement each term in the Boolean expression as one slice of the pattern-match engine. A slice consists of an input

selector and detection logic that continuously monitors the selected input and creates a HIGH output (true). If the input qualifies as detected, several terms combine to a minterm. Assertion of a pin interrupt occurs when the minterm evaluates as true.

The detection logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detection logic output remains high) until the pattern-match engine detect logic is cleared again.
- Event (nonsticky): Every time an edge (rising or falling) is detected, the detection logic output goes HIGH. After one clock cycle, writing 1 to this bit enables the detection logic to detect another edge.
- Level: A high or low level on the selected input.

The following figure shows the details of the edge-detection logic for each slice.

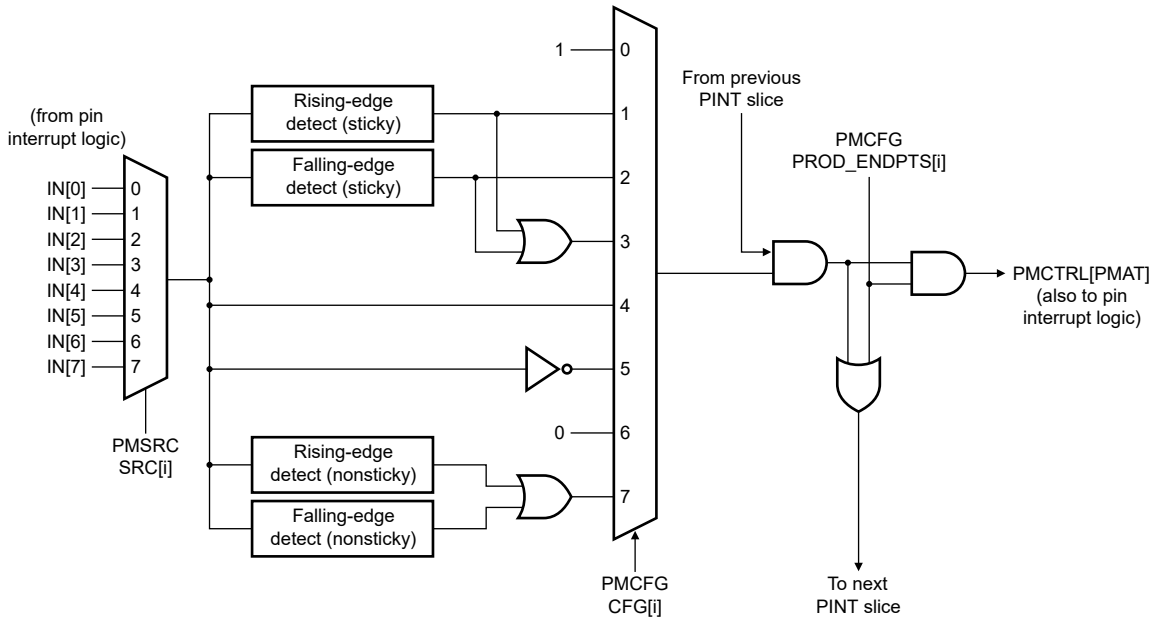


Figure 439. Pattern-match bit slice

Combined sticky and nonsticky events form a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

Combining a level detect with an event detect can create a time window during which rising or falling edges will create a pin interrupt. See Pattern-match engine edge-detect examples for details.

The following figure shows connections between the pins and the pattern-match engine. The INPUTMUX block selects all pins that are inputs to the pattern-match engine and can be GPIO port pins or other pin functions depending on the configuration.

NOTE

The pattern-match feature requires clocks in order to operate. It cannot generate an interrupt or wake the chip during reduced power modes below sleep mode.

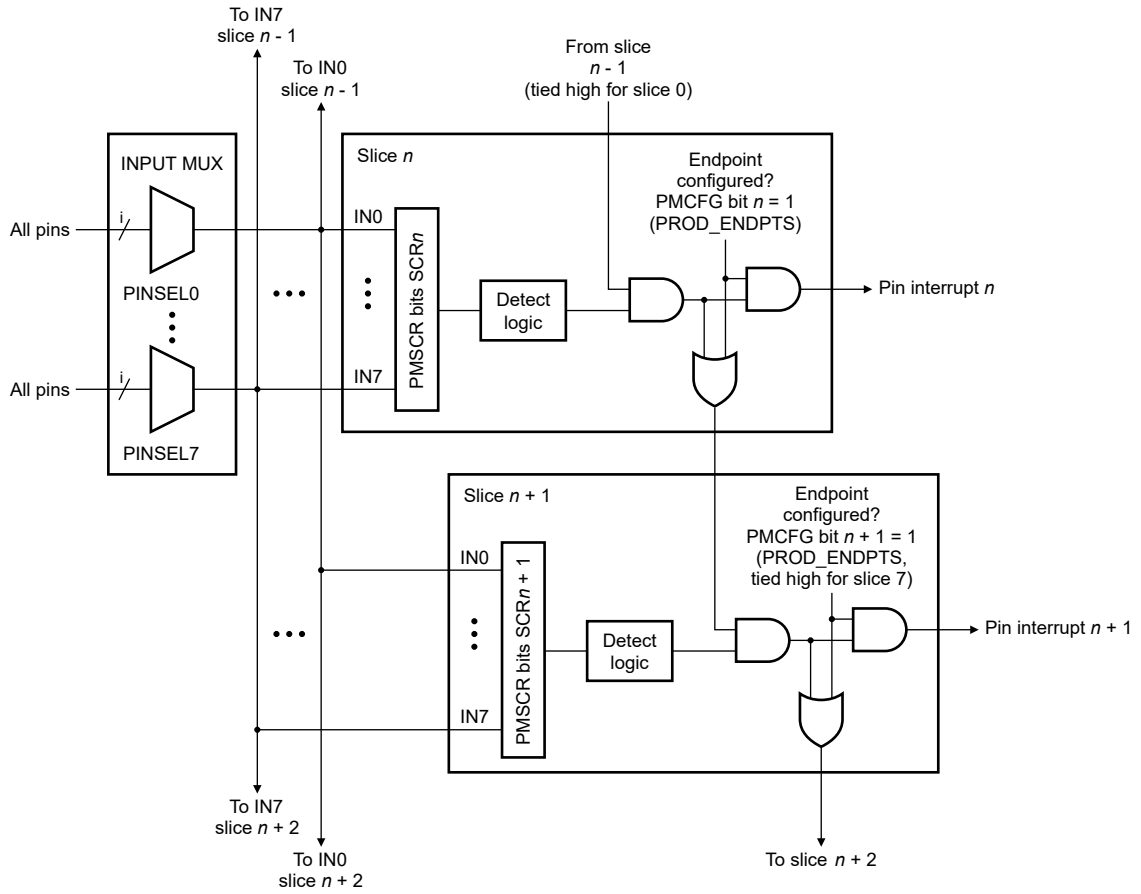


Figure 440. Pattern-match engine connections

The pattern-match logic continuously monitors the 8 inputs and generates interrupts when any one or more minterms (product terms) of the specified Boolean expression match. Each individual minterm generates a separate interrupt request.

It is also possible to enable the pattern-match module to generate RXEV output if any minterm in the Boolean expression matches.

The pattern-match function uses the same 8 interrupt request lines as the pin interrupts. Therefore, in terms of interrupt generation, these two features are mutually exclusive. Control bit generates interrupt requests based on standard pin interrupts or pattern matches. When selecting the pin interrupts, enable the RXEV request to the CPU for pattern matches.

NOTE

Do not use pattern matching to wake the part from deep-sleep mode. Pin interrupts must be selected in order to use the GPIO for wake-up.

8 bit-slice elements make up the pattern-match module. Each bit slice represents a single component in a minterm (product term) within the Boolean expression. Every time the last bit slice for a particular minterm matches, an interrupt request is asserted.

Use the pattern-match capability to create complex software state machines. Each minterm (and its corresponding individual interrupt) represents a different transition event to a new state. Then establish the new set of conditions (that is a new Boolean expression) to make a transition out of the current state.

74.3.2 Clocking

The PINT module has a single clock input, the APB PCLK.

This clock is used for its APB interface, as well as all other peripheral functionality. It must be enabled by the system for register accesses and pattern matching.

74.3.3 Interrupts

The PINT module has 8 interrupt channels, which can be configured by its registers. Each channel has its own interrupt output. Depending on how this IP is integrated, the interrupt outputs can be ORed together to use a single system-level interrupt.

74.4 External signals

PINSELa determines the inputs to the pin interrupt and pattern-match and engines. See the "INPUTMUX" chapter for more details.

74.5 Initialization

For initialization steps, see chip-specific section.

74.6 Application information

Pattern-Match Interrupt Bit-Slice Source (PMSRC) and Pattern-Match Interrupt Bit Slice Configuration (PMCFG) specify the following expression:

```
IN0 AND NOT IN1 AND IN3 rising edge OR IN1 AND IN2 OR IN0 AND NOT IN3 AND NOT IN4
```

Each term in the Boolean expression, IN0, NOT IN1, IN3 rising edge, and so on, represents one bit slice of the pattern-match engine.

- In the first AND function IN0 AND NOT IN1 AND IN3 rising edge, bit slice 0 monitors for a high level on input IN0, bit slice 1 monitors for a low level on input IN1, and bit slice 2 monitors for a rising edge on input IN3. Detecting this combination of features (all three terms are true) asserts the interrupt associated with bit slice 2.
- In the second AND function IN1 AND IN2, bit slice 3 monitors input IN1 for a high level and bit slice 4 monitors input IN2 for a high level. Detecting this combination asserts the interrupt associated with bit slice 4.
- In the third AND function IN0 AND NOT IN3 AND NOT IN4, bit slice 5 monitors input IN0 for a high level, bit slice 6 monitors input IN3 for a low level, and bit slice 7 monitors input IN4 for a low level. Detecting this combination asserts the interrupt associated with bit slice 7.
- The ORed result of all three AND functions asserts the RXEV request to the CPU. That is, any of the three terms that are true will assert the output.

74.6.1 Pattern-match engine example

Suppose you want to match the Boolean pattern

(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

with:

IN6fe = (sticky) falling edge on input 6

IN7ev = (nonsticky) event (rising or falling edge) on input 7

Each individual term in the above expression is controlled by one bit slice. To specify this expression, program the pattern-match bit slice source and configuration register fields as follows:

- Pattern-Match Interrupt Bit-Slice Source (PMSRC)
  - Because bit slice 5 is used to detect a sticky event on input 6, you can write 1 to SRC5 to clear any preexisting edge detects on bit slice 5.
  - SRC0: 001 - select input 1 for bit slice 0
  - SRC1: 001 - select input 1 for bit slice 1
  - SRC2: 010 - select input 2 for bit slice 2

- SRC3: 010 - select input 2 for bit slice 3
- SRC4: 011 - select input 3 for bit slice 4
- SRC5: 110 - select input 6 for bit slice 5
- SRC6: 101 - select input 5 for bit slice 6
- SRC7: 111 - select input 7 for bit slice 7

- Pattern-Match Interrupt Bit Slice Configuration (PMCFG)
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS2 = 1
  - PROD_ENDPTS5 = 1
  - All other slices are not product term endpoints and their PMCFG[PROD_ENDPTSn] are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
  - PROD_ENDPTS= 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
  - CFG0: 000 - high level on the selected input (input 1) for bit slice 0
  - CFG1: 000 - high level on the selected input (input 1) for bit slice 1
  - CFG2: 000 - high level on the selected input (input 2) for bit slice 2
  - CFG3: 101 - low level on the selected input (input 2) for bit slice 3
  - CFG4: 101 - low level on the selected input (input 3) for bit slice 4
  - CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
  - CFG6: 000 - high level on the selected input (input 5) for bit slice 6
  - CFG7: 111 - event (any edge, nonsticky) on the selected input (input 7) for bit slice 7
- Pattern-Match Interrupt Control (PMCTRL)
  - Bit0: Writing this bit selects pattern matches to generate the pin interrupts in place of the standard pin interrupt mechanism.

    For this example, detecting a match on the first product term asserts pin interrupt 0 (which, in this case, is just a high level on input 1).

    Pin interrupt 2 is asserted in response to a match on the second product term.

    Pin interrupt 5 is asserted when there is a match on the third product term.

    Pin interrupt 7 is asserted on a match on the last term.
  - Bit1: Writing this bit causes the RXEV signal to the CPU getting asserted whenever a match occurs on any of the product terms in the expression. Otherwise, the RXEV line does not get used.
  - Bit31:24: At any given time, bits 0, 2, 5, and 7 may be high if the corresponding product terms are currently matching.
  - The remaining bits are low.

74.6.2 Pattern-match engine edge-detect examples

The following figures show only pattern-match functionality in which accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

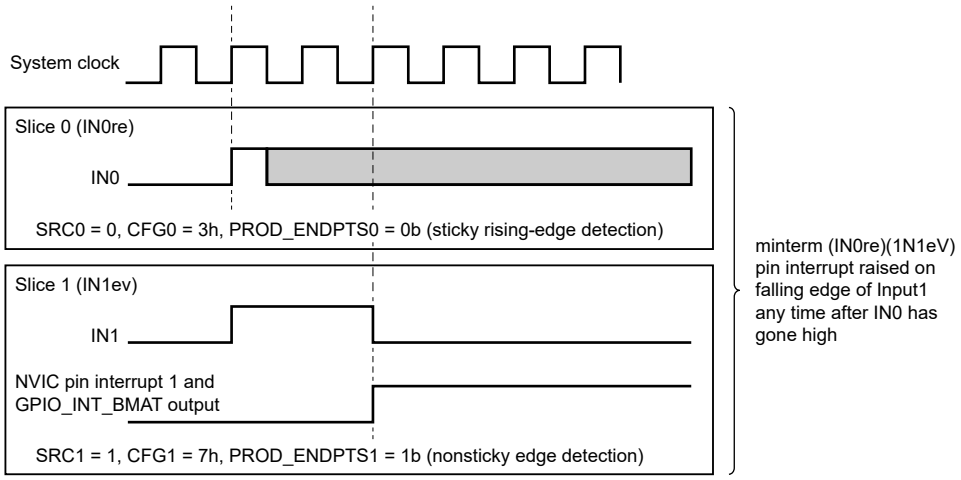


Figure 441. Pattern-match engine examples: sticky edge detect

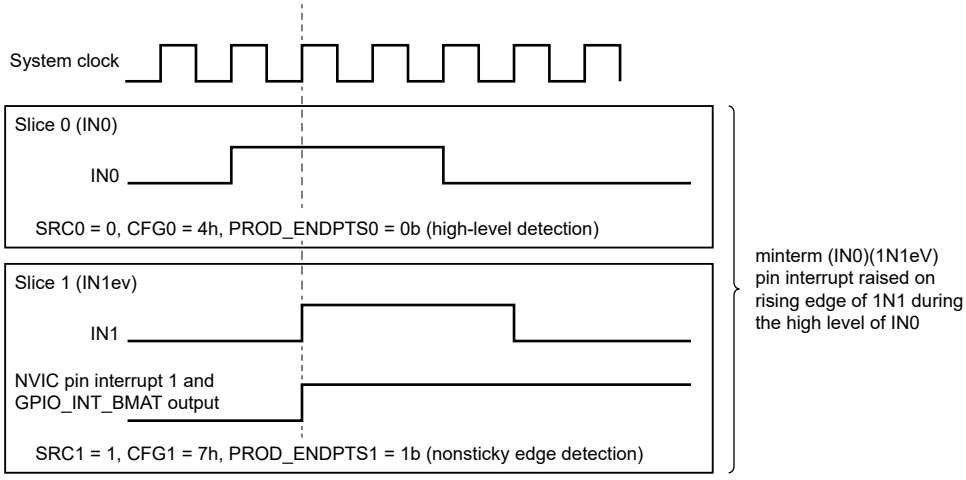


Figure 442. Pattern-match engine examples: Windowed nonsticky edge-detect evaluates as true

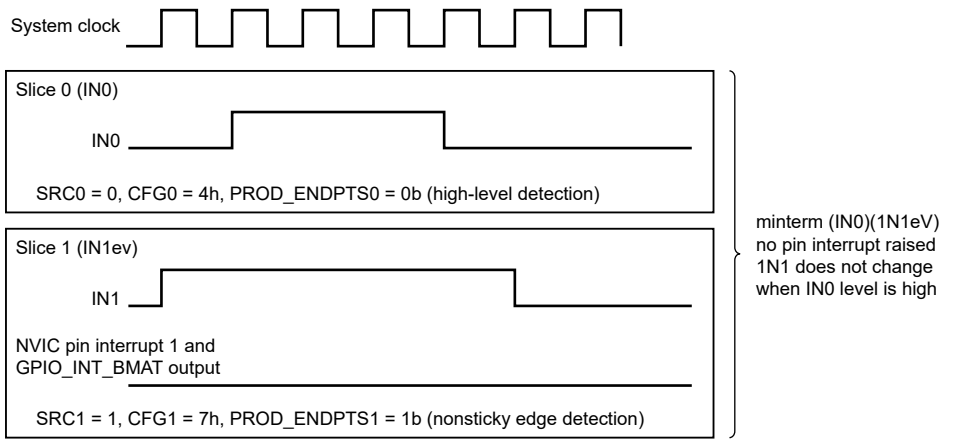


Figure 443. Pattern-match engine examples: Windowed nonsticky edge-detect evaluates as false

74.7 Memory map and register definition

This section includes the PINT memory map and detailed descriptions of all registers.

The chip includes up to 8 pins identified as interrupt sources by Pin Interrupt Select (PINSELa). The registers have 8 fields that correspond to the pins called out by PINSEL.

Table 712. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high active.
CIENF	Write to disable falling-edge interrupts.	Write to select low active.

74.7.1 Pin Interrupts and Pattern Match register descriptions

74.7.1.1 PINT memory map

PINT0 base address: 4000_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Pin Interrupt Mode (ISEL)	32	RW	See section
4h	Pin Interrupt Level or Rising-Edge Interrupt Enable (IENR)	32	RW	See section
8h	Pin Interrupt Level or Rising-Edge Interrupt Set (SIENR)	32	W	See section
Ch	Pin Interrupt Level (Rising-Edge Interrupt) Clear (CIENR)	32	RW	See section
10h	Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF)	32	RW	See section
14h	Pin Interrupt Active Level or Falling-Edge Interrupt Set (SIENF)	32	W	See section
18h	Pin Interrupt Active Level or Falling-Edge Interrupt Clear (CIENF)	32	W	See section
1Ch	Pin Interrupt Rising Edge (RISE)	32	RW	See section
20h	Pin Interrupt Falling Edge (FALL)	32	RW	See section
24h	Pin Interrupt Status (IST)	32	RW	See section
28h	Pattern-Match Interrupt Control (PMCTRL)	32	RW	See section
2Ch	Pattern-Match Interrupt Bit-Slice Source (PMSRC)	32	RW	See section
30h	Pattern-Match Interrupt Bit Slice Configuration (PMCFG)	32	RW	See section

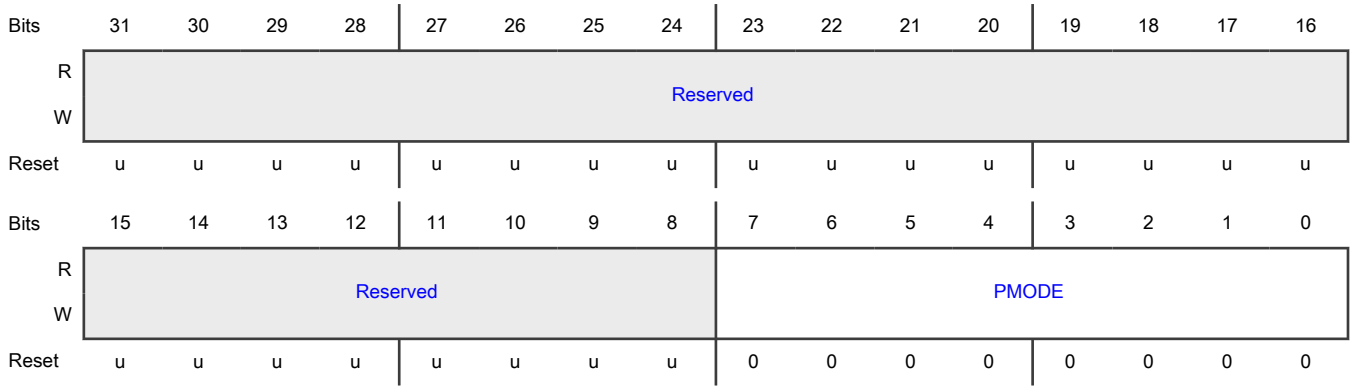
74.7.1.2 Pin Interrupt Mode (ISEL)

Offset

Register	Offset
ISEL	0h

Function
Determines whether an interrupt is edge- or level-sensitive.

Diagram



Fields

Field	Function
31-8 —	Read value is undefined; write only 0.
7-0 PMODE	Interrupt mode Selects the interrupt mode for each pin interrupt. Bit *n* configures the pin interrupt selected in PINSEL in INPUTMUX. 0000_0000b - In bit n configures the interrupt to be edge-sensitive 0000_0001b - In bit n configures the interrupt to be level-sensitive

74.7.1.3 Pin Interrupt Level or Rising-Edge Interrupt Enable (IENR)

Offset

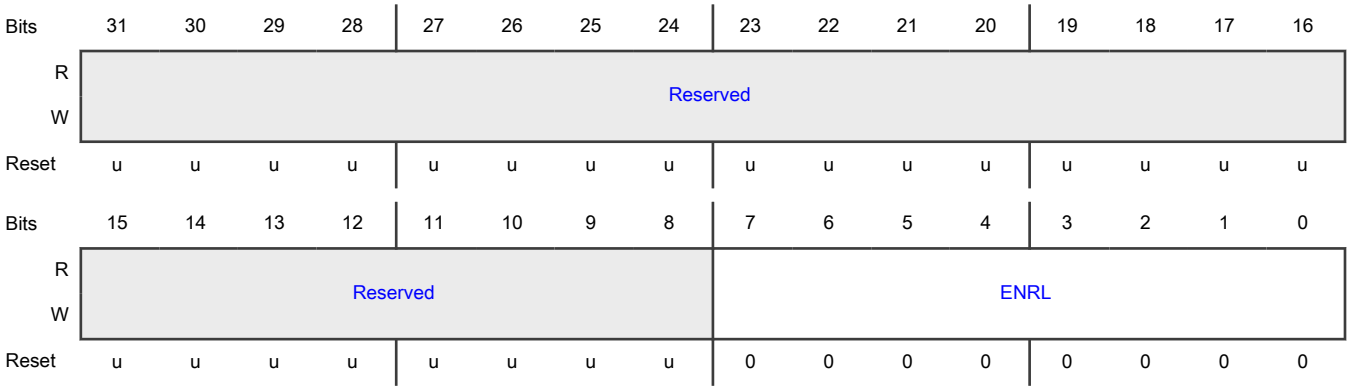
Register	Offset
IENR	4h

Function

Enables the interrupt by each field in Pin Interrupt Level or Rising-Edge Interrupt Enable (IENR), depending on the ISEL[PMODE] configured in Pin Interrupt Mode (ISEL).

- If the PMODE is edge sensitive (PMODE = 0), the rising-edge interrupt is enabled.
- If the PMODE is level sensitive (PMODE = 1), the level interrupt is enabled. Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF) configures the active level (HIGH or LOW) for this interrupt.

Diagram



Fields

Field	Function
31-8 —	Read value is undefined; write only 0.
7-0 ENRL	Enables Interrupt Enables the rising edge or level interrupt for each pin interrupt corresponding to the bit index. Bit *n* configures the pin interrupt selected in PINSELa. 0000_0000b - In bit n disables the corresponding interrupt 0000_0001b - In bit n enables the corresponding interrupt

74.7.1.4 Pin Interrupt Level or Rising-Edge Interrupt Set (SIENR)

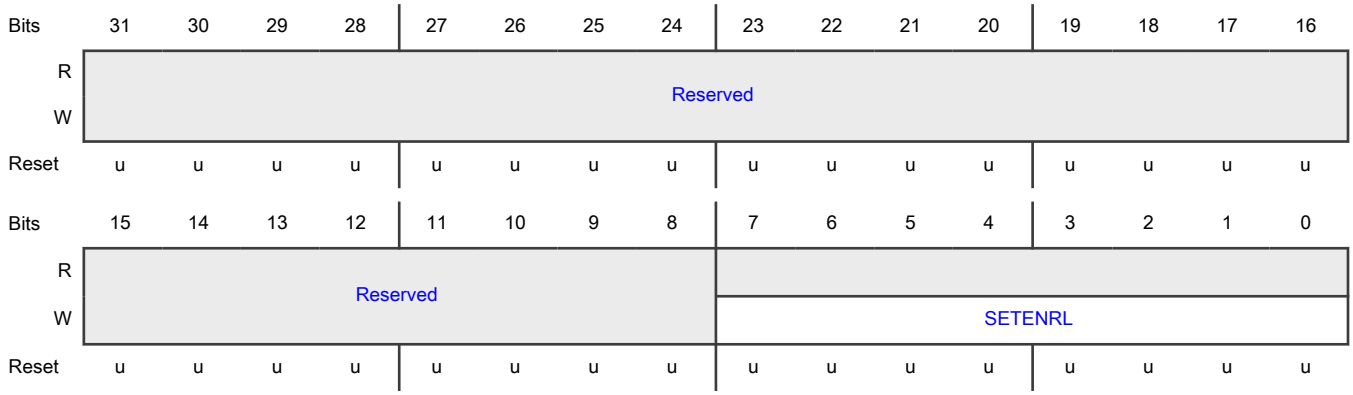
Offset

Register	Offset
SIENR	8h

Function

Configures the corresponding field in Pin Interrupt Level or Rising-Edge Interrupt Enable (IENR) by each field in Pin Interrupt Level or Rising-Edge Interrupt Set (SIENR), depending on the ISEL[PMODE] configured in Pin Interrupt Mode (ISEL).

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SETENRL	Configures IENR Configures IENR by writing 1s to this address. Bit *n* configures the corresponding bit in IENR. 0000_0000b - No operation for interrupt n 0000_0001b - Enable rising edge or level interrupt for interrupt n

74.7.1.5 Pin Interrupt Level (Rising-Edge Interrupt) Clear (CIENR)

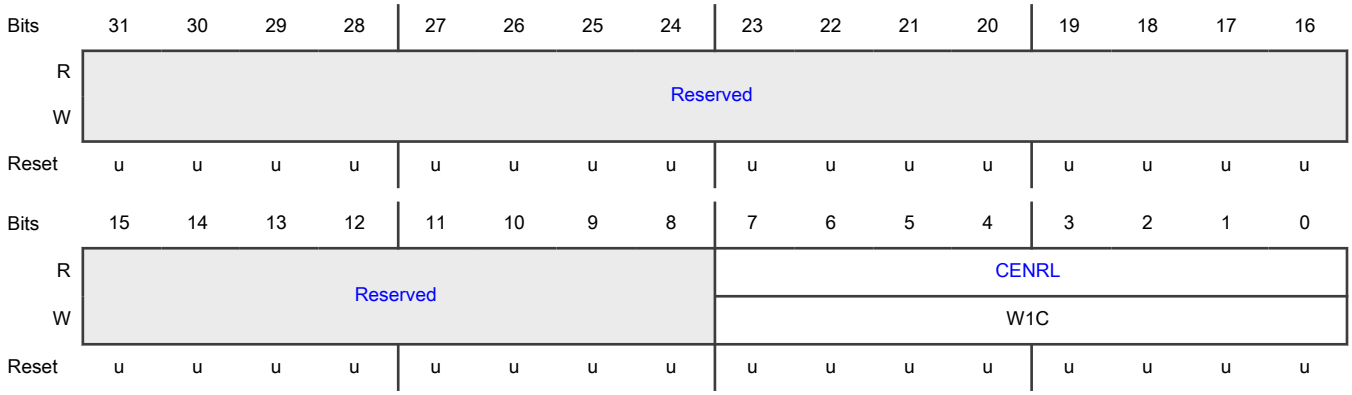
Offset

Register	Offset
CIENR	Ch

Function

Writing 0 to each bit in Pin Interrupt Level (Rising-Edge Interrupt) Clear (CIENR) adjusts the corresponding bit in Pin Interrupt Level or Rising-Edge Interrupt Enable (IENR) depending on the ISEL[PMODE] configured in Pin Interrupt Mode (ISEL). Writing 0 to this address configures bits in Pin Interrupt Level or Rising-Edge Interrupt Enable (IENR), effectively disabling interrupts. Writing 0 to bit *a* configures bit *n* in Pin Interrupt Level or Rising-Edge Interrupt Enable (IENR).

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 CENRL	Clear bits in IENR 0000_0000b - No operation 0000_0001b - Disable rising edge or level interrupt

74.7.1.6 Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF)

Offset

Register	Offset
IENF	10h

Function

Enables the falling-edge interrupt or configures the level sensitivity depending on the ISEL[PMODE] configured in Pin Interrupt Mode (ISEL).

- If the PMODE is edge sensitive (PMODE = 0), the falling-edge interrupt is enabled.
- If the PMODE is level sensitive (PMODE = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								ENAF							
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 ENAF	Enables Interrupt Enables the falling edge or configures the active-level interrupt for each pin interrupt. Bit *a* configures the pin interrupt selected in PINSELa. 0000_0000b - Disable (set active interrupt level LOW) 0000_0001b - Enable (set active interrupt level HIGH)

74.7.1.7 Pin Interrupt Active Level or Falling-Edge Interrupt Set (SIENF)

Offset

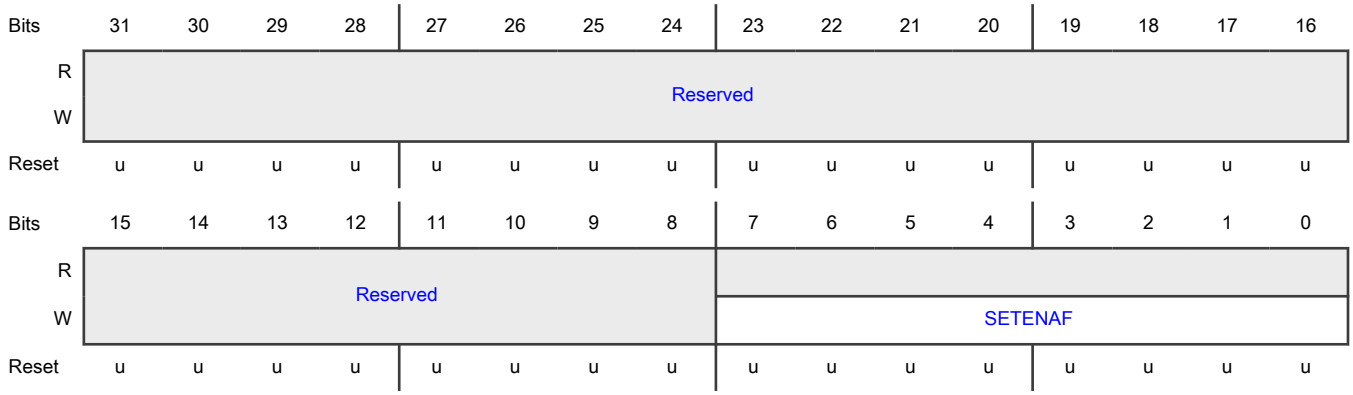
Register	Offset
SIENF	14h

Function

Sets the corresponding bit in Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF) (by each bit of SIENF), depending on the PMODE configured in Pin Interrupt Mode (ISEL).

- If the PMODE is edge sensitive (PMODE = 0), the falling-edge interrupt becomes 1.
- If the PMODE is level sensitive (PMODE = 1), the HIGH-active interrupt is selected.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SETENAF	Write 1 to this address to clear Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF) to disable interrupts. Bit *a* sets bit *n* in IENF. 0000_0000b - Writes 0 to IENF. No operation 0000_0001b - Select HIGH-active interrupt or enable falling-edge interrupt

74.7.1.8 Pin Interrupt Active Level or Falling-Edge Interrupt Clear (CIENF)

Offset

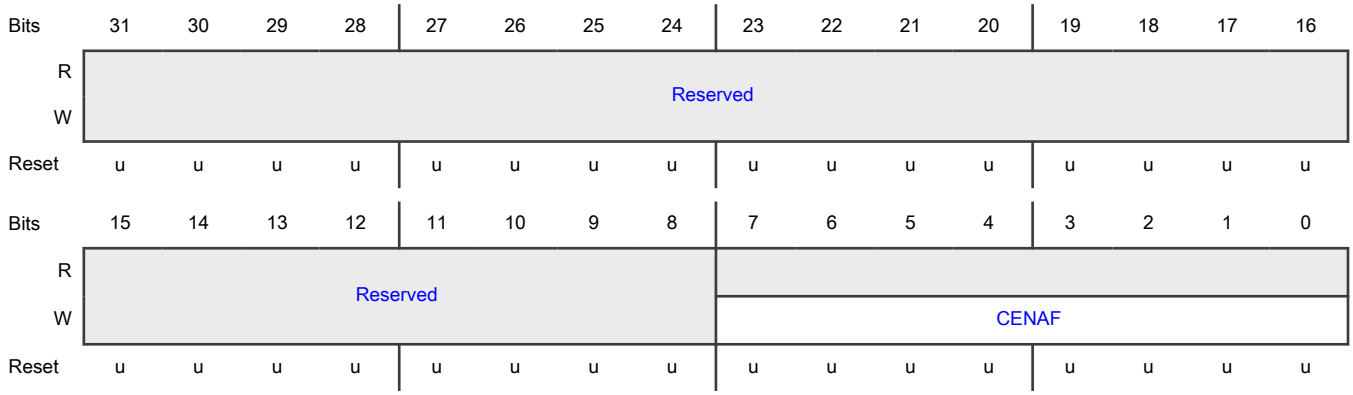
Register	Offset
CIENF	18h

Function

Sets the corresponding bit in Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF) (by each bit of Pin Interrupt Active Level or Falling-Edge Interrupt Clear (CIENF), depending on the ISEL[PMODE] configured in Pin Interrupt Mode (ISEL).

- If the PMODE is edge sensitive (PMODE = 0), the falling-edge interrupt is cleared.
- If the PMODE is level sensitive (PMODE = 1), the LOW-active interrupt is selected.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 CENAF	Writes 0 to IENF Specifies that if you change Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF) to 0 by writing 1 to it,this will disable interrupts. Writing 0 to bit *a* configures bit *n* in Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF). 0000_0000b - No operation 0000_0001b - LOW-active interrupt selected or falling-edge interrupt disabled

74.7.1.9 Pin Interrupt Rising Edge (RISE)

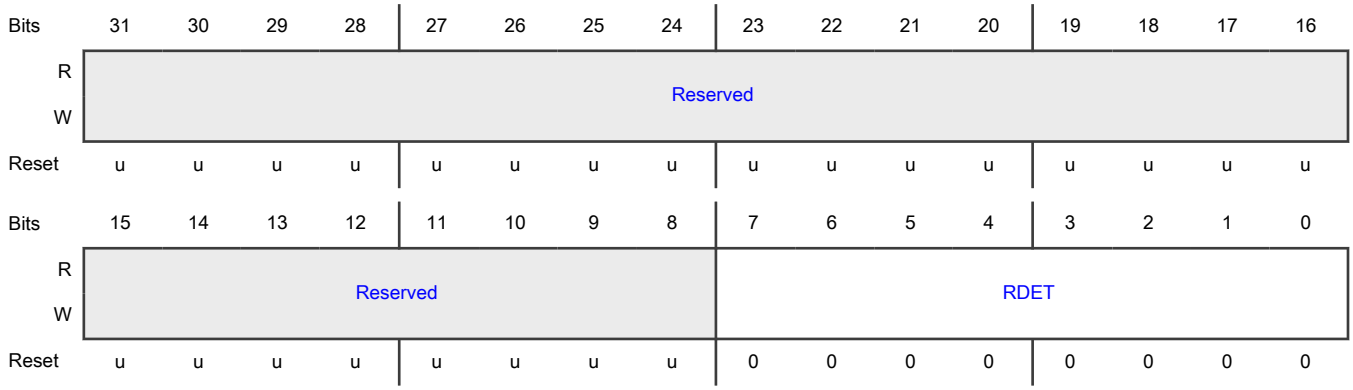
Offset

Register	Offset
RISE	1Ch

Function

Contains 1s for pin interrupts selected in PINSELa (see the "INPUTMUX" chapter for more details) on which a rising edge has been detected. Writing 1s to this register clears rising-edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by PINSEL, regardless of whether they are interrupt-enabled.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 RDET	Rising-Edge Detect Bit *a* detects the rising edge of the pin selected in PINSELa. 0000_0000b - Read 0- No rising edge (since Reset or you wrote a 1 to this field last time), Write 0- No operation 0000_0001b - Read 1- Rising edge (since Reset or you wrote a 1 to this field last time), Write 1- Clear rising-edge detection for this pin

74.7.1.10 Pin Interrupt Falling Edge (FALL)

Offset

Register	Offset
FALL	20h

Function

Contains 1s for pin interrupts selected in PINSELa on which a falling edge has been detected. Writing 1s to this register clears falling-edge detection. The values (1s) in this register assert interrupt requests for pins with falling-edge interrupt capabilities. PINSELa detects all edges for all pins, regardless of whether they are interrupt-enabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								FDET							
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7-0 FDET	Falling-Edge Detect Bit *a* detects the falling edge of the pin selected in PINSELa. 0000_0000b - Read 0- No falling edge (since Reset or you wrote a 1 to this field last time), Write 0- No operation 0000_0001b - Read 1- Falling edge (since Reset or you wrote a 1 to this field last time), Write 1- Clear falling-edge detection for this bit

74.7.1.11 Pin Interrupt Status (IST)

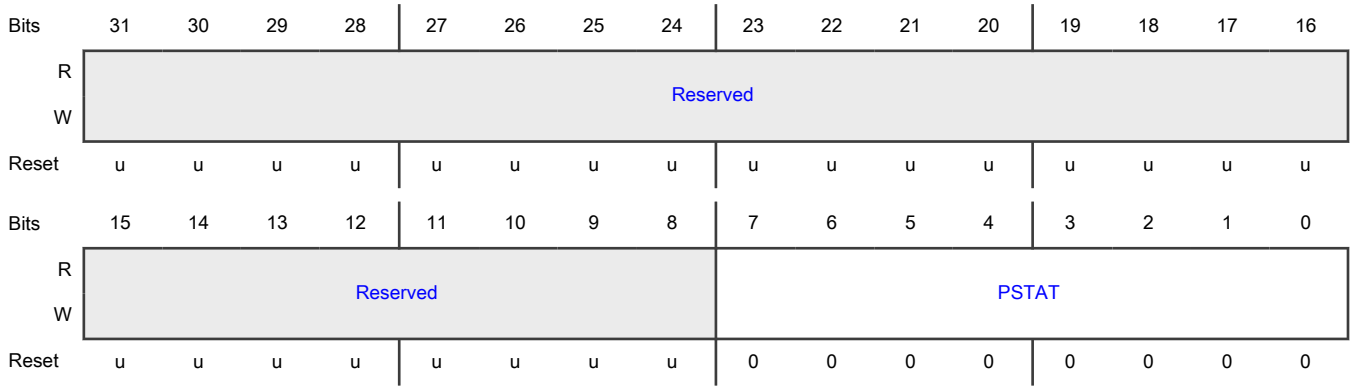
Offset

Register	Offset
IST	24h

Function

Returns 1s for pin interrupts that are currently requesting an interrupt. For edge-sensitive pins in Interrupt Select, writing 1s to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing 1s inverts the corresponding bit in the active-level register, thus switching the active level on the pin.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 PSTAT	Pin Interrupt Status Bit *a* returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINSELa. 0000_0000b - Read 0- Interrupt is not requested, Write 0- No operation 0000_0001b - Read 1- Interrupt is requested, Write 1 (edge-sensitive)- clear rising- and falling-edge detection for this pin, Write 1 (level-sensitive)- switch the active level for this pin in Pin Interrupt Active Level or Falling-Edge Interrupt Enable (IENF)

74.7.1.12 Pattern-Match Interrupt Control (PMCTRL)

Offset

Register	Offset
PMCTRL	28h

Function

Contains a field to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. Read the current state of the pattern match in this register. If the pattern-match feature is not used (either for interrupt generation or for RXEV assertion), write 0 to PMCTRL[SEL_PMATCH] and PMCTRL[ENA_RXEV] of this register to conserve power.

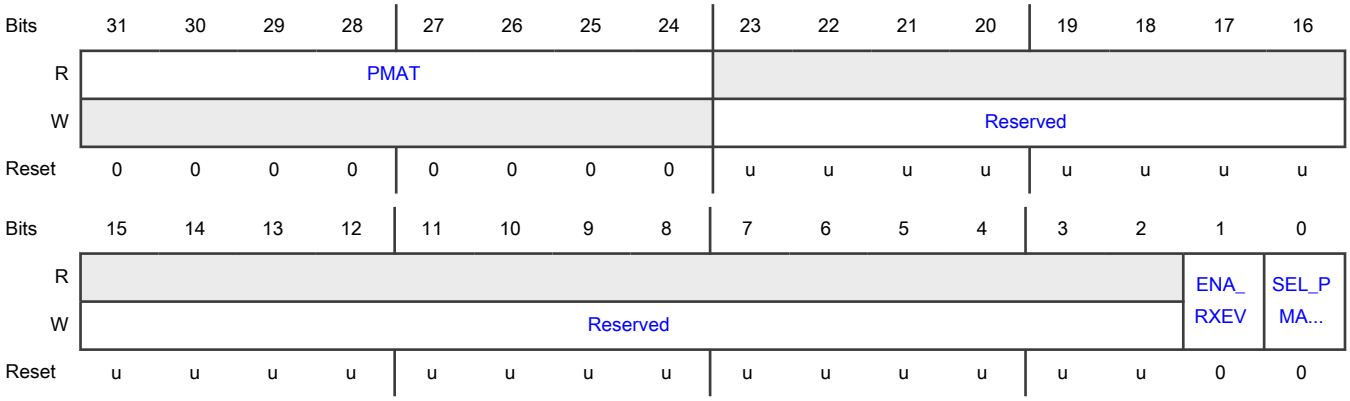
NOTE

Set up the pattern-match configuration in Pattern-Match Interrupt Bit-Slice Source (PMSRC) and Pattern-Match Interrupt Bit Slice Configuration (PMCFG) before writing to this register to enable (or reenable) the pattern-match functionality. This prevents spurious interrupts from occurring during the feature"s activation.

NOTE

The pattern-match feature requires clocks to operate and thus not generate an interrupt or wake the chip during reduced power modes below sleep mode.

Diagram



Fields

Field	Function
31-24 PMAT	Pattern Matches Displays the current state of pattern matches. 0000_0001b - The corresponding product term is matched by the current state of the appropriate inputs
23-2 —	Reserved, will read 0"s
1 ENA_RXEV	Enables the RXEV output to the CPU and/or to a GPIO output, when the specified Boolean expression evaluates to true. If this value is 0b, RXEV output to the CPU is disabled. If this value is 1b, RXEV output to the CPU is enabled. 0b - Disabled 1b - Enabled
0 SEL_PMATCH	Specifies whether the pin interrupts are controlled by the pin interrupt function or by the pattern-match function. If this value is 0b, interrupts are driven in response to the standard pin interrupt function. If this value is 1b, interrupts are driven in response to pattern matches. 0b - Pin interrupt 1b - Pattern match

74.7.1.13 Pattern-Match Interrupt Bit-Slice Source (PMSRC)

Offset

Register	Offset
PMSRC	2Ch

Function

Specifies the input source for each of the pattern-match bit slices. Each of the possible 8 inputs is selected in the pin interrupt select registers in INPUTMUX. Input 0 corresponds to the pin selected in PINSEL0, input 1 corresponds to the pin selected in PINSEL1, and so on.

NOTE

Writing any value to either Pattern-Match Interrupt Bit Slice Configuration (PMCFG) or Pattern-Match Interrupt Bit-Slice Source (PMSRC), or disabling the pattern-match feature by clearing both the PMCTRL[SEL_PMATCH] and PMCTRL[ENA_RXEV] bits in Pattern-Match Interrupt Control (PMCTRL) to zeros will erase all edge-detect history.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SRC7				SRC6				SRC5				SRC4			SRC3				SRC2
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SRC2		SRC1			SRC0			Reserved							
Reset	0	0	0	0	0	0	0	0	u	u	u	u	u	u	u	u

Fields

Field	Function
31-29: SRC7 28-26: SRC6 25-23: SRC5 22-20: SRC4 19-17: SRC3 16-14: SRC2 13-11: SRC1 10-8: SRC0	Selects the input source for bit slice n 000b - Input 0 (selects the pin identified in PINSEL0) 001b - Input 1 (selects the pin identified in PINSEL1) 010b - Input 2 (selects the pin identified in PINSEL2) 011b - Input 3 (selects the pin identified in PINSEL3) 100b - Input 4 (selects the pin identified in PINSEL4) 101b - Input 5 (selects the pin identified in PINSEL5) 110b - Input 6 (selects the pin identified in PINSEL6) 111b - Input 7 (selects the pin identified in PINSEL7)
7-0 —	Reserved, will read 0's

74.7.1.14 Pattern-Match Interrupt Bit Slice Configuration (PMCFG)

Offset

Register	Offset
PMCFG	30h

Function

Configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit slices are the end-points of product terms in the Boolean expression (where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

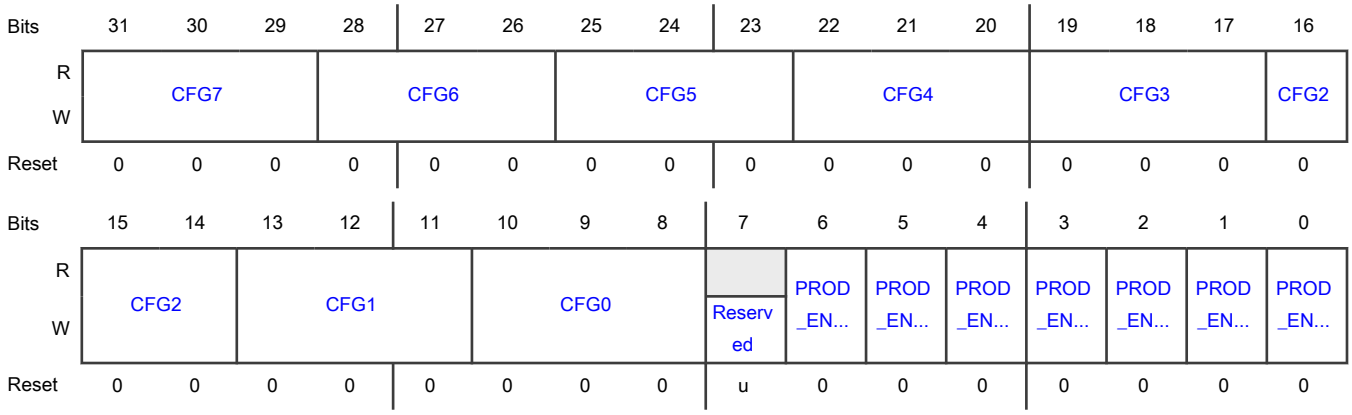
- Sticky: A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism becomes 0. The input qualifies as detected (the detect logic output remains HIGH) until the pattern-match engine detect logic becomes 0 again.
- Nonsticky: In the event of an edge (rise or fall), the detect logic output for this pin goes HIGH. One clock cycle clears this bit, allowing the edge detector to detect a second edge.

To clear the pattern-match engine detect logic, write any value to either Pattern-Match Interrupt Bit Slice Configuration (PMCFG) or Pattern-Match Interrupt Bit-Slice Source (PMSRC), or disable the pattern-match feature by changing both PMCTRL[SEL_PMATCH] and PMCTRL[ENA_RXEV] to 0 by writing 1 to Pattern-Match Interrupt Control (PMCTRL). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the Boolean expression, write a 1 in the corresponding PROD_ENDPTSa bit. Setting a term as the final component has two effects:

1. Anytime a match to that product term occurs, this leads to an assertion of interrupt request associated with this bit slice.
2. The next bit slice will start a new, independent product term in the Boolean expression. The Boolean expression will include an OR following the element controlled by this bit slice.

Diagram



Fields

Field	Function
31-29: CFG7 28-26: CFG6 25-23: CFG5 22-20: CFG4 19-17: CFG3 16-14: CFG2 13-11: CFG1 10-8: CFG0	Match Configuration Specifies the match contribution condition for bit slice n: • If the value is 000b, this bit slice always contributes to a product term match. • If the value is 001b, edge detection for this bit slice is clear when there has been a rising edge on the specified input since last clearing. In order to clear this match condition, you must write to Pattern-Match Interrupt Bit Slice Configuration (PMCFG) or Pattern-Match Interrupt Bit-Slice Source (PMSRC). • If the value is 010b, edge detection for this bit slice is clear when there has been a falling edge on the specified input since last clearing. In order to clear this match condition, you must write to Pattern-Match Interrupt Bit Slice Configuration (PMCFG) or Pattern-Match Interrupt Bit-Slice Source (PMSRC). • If the value is 011b, edge detection for this bit slice is clear when there has been either a rising or a falling edge on the specified input since last clearing. In order to clear this match condition, you must write to Pattern-Match Interrupt Bit Slice Configuration (PMCFG) or Pattern-Match Interrupt Bit-Slice Source (PMSRC). • If the value is 100b, match for this bit slice occurs when there is a high level on the input specified for this bit slice in Pattern-Match Interrupt Bit-Slice Source (PMSRC). • If the value is 101b, match occurs when there is a low level on the specified input. • If the value is 110b, it never contributes to a match and must be used to disable any unused bit slices. • If the value is 111b, a match occurs when the specified input first detects a rising or falling edge. This is a nonsticky version of value 0 x 3. Write 0 to this field after one clock cycle. 000b - Constant HIGH 001b - Sticky rising edge 010b - Sticky falling edge 011b - Sticky rising or falling edge 100b - High level 101b - Low level 110b - Constant 0 111b - Event (Nonsticky rising or falling edge)
7 —	Bit slice 7 is automatically considered a product endpoint.
6-0 PROD_ENDPTSn	Determines whether slice n is an endpoint. Slice n is not an endpoint. Slice n is the endpoint of a product term (minterm). Pin interrupt n in the NVIC is raised if the minterm evaluates as true. 0b - No effect 1b - Endpoint

Chapter 75
Port Control (PORT)

75.1 Chip-specific PORT information

Table 713. Reference links to related information

Topic	Related module	Reference
Full description	PORT	PORT
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

See Peripheral Reset Control 0 (PRESETCTRL0) to know how to reset PORT. To enable PORT clock, see AHB Clock Control 0 (AHBCLKCTRL0).

75.1.1 Module instances

This device has six instances of the port module, PORT0, PORT1, PORT2, PORT3, PORT4, and PORT5.

NOTE

For PORT5, the EFT detect function can only work when VBAT = 1.8 V.

75.1.2 Normal I/O pin structure

The following diagram shows the structure of normal I/O pin.

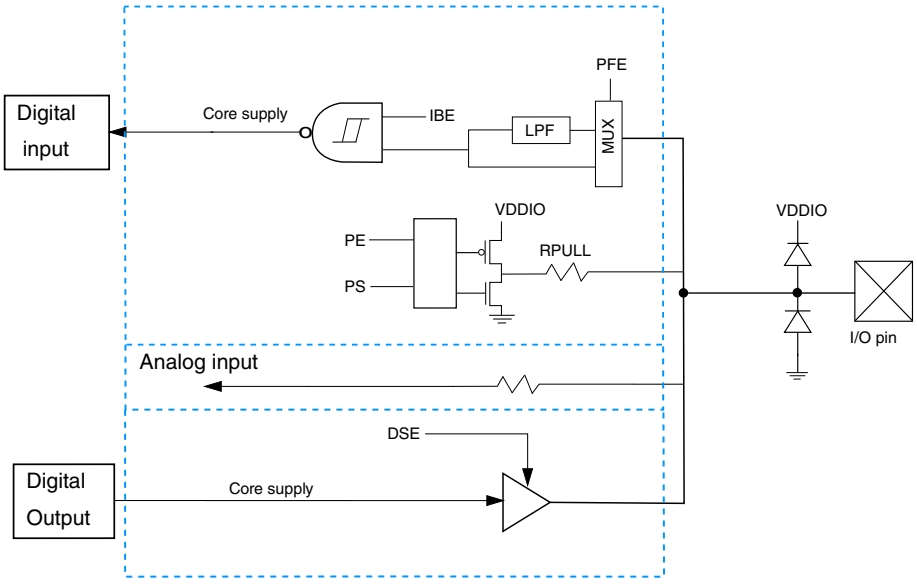


Figure 444. Basic structure of an I/O port

75.1.3 Security considerations

For secure applications, NXP recommends that this module is configured for secure and privileged access only to prevent unintentional or malicious modification of the system operation by nonsecure software. The secure AHB controller controls the security level for access to peripherals and does default to secure and privileged access for all peripherals.

75.2 Overview

PORT provides support for pad control functions. You can configure most functions independently for each pin, in the 32-bit port, and affect the pin regardless of its pin multiplexing state.

There exists a single instance of the PORT module for each port, and not all pins within each port are implemented on a specific chip.

75.2.1 Block diagram

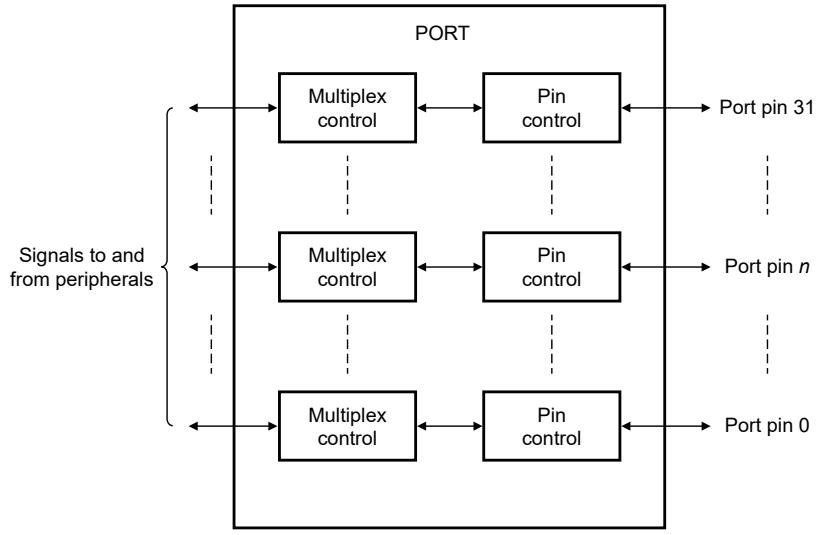


Figure 445. Block diagram

75.2.2 Features

- Individual pull control fields with pullup, pulldown, and pull-disable support
- Individual PCR*n*[PFE] fields that enable and disable individual input passive filters
- Individual PCR*n*[ODE] fields that enable and disable individual open drain outputs
- Digital input inversion to optionally invert the digital input
- Individual electrical fast transient (EFT) detect with an EFT detect flag and associated interrupt
- Digital PCR*n*[IBE] fields to configure between analog or disabled functions and digital functions
- Individual PCR*n*[MUX] fields supporting GPIO and up to 3 chip-specific digital functions

75.3 Functional description

75.3.1 Pin control

Each port pin has a corresponding Pin Control Register (PCR) associated with it that helps you configure the following functions for each pin within the 32-bit port:

- Pullup or pulldown enable
- Open drain enable
- Passive input filter enable
- Digital input inversion
- EFT detection
- Software configuration lock
- Pin multiplexing mode

These functions apply across all digital pin multiplexing modes, and individual peripherals do not override the configuration in PCR*n* unless otherwise noted. For example, if an I^2C function is enabled on a pin, it does not override the pullup or open drain configuration for that pin.

PCR*n*[LK] allows the configuration for each pin to be locked until the next system reset. When locked, writes to the lower half of that PCR*n* are ignored, although a bus error is not generated on an attempted write to a locked register.

The configuration of each PCR*n* is retained when the PORT module is disabled.

When you configure a pin in a digital pin multiplexing mode, the input buffer for that pin is enabled, allowing the pin state to be read via the corresponding GPIO.PDIR or allowing a pin interrupt or DMA request to be generated. If a pin is always floating when its input buffer is enabled, it can cause an increase in power consumption. This situation must be avoided. A pin can be floating because of an input pin that is not connected or an output pin that is tristated (output buffer is disabled).

Enabling the internal pull resistor (or implementing an external pull resistor) ensures that a pin does not float when its input buffer is enabled. The internal pull resistor is automatically disabled whenever the output buffer is enabled, allowing PCR*n*[PE] to remain 1. Configuring Pin Multiplexing mode to disabled or analog (PCR*n*[MUX] = 0) disables the pin's input buffer and results in lowest power consumption.

75.3.2 Global pin control

Global Pin Control Low (GPCLR) and Global Pin Control High (GPCHR) allow a single register write to update the lower 16 bits of PCR*n* for up to 16 pins, all with the same value. You cannot write to locked registers by using Global Pin Control Low (GPCLR) and Global Pin Control High (GPCHR).

Global Pin Control Low (GPCLR) and Global Pin Control High (GPCHR) enable you to quickly configure multiple pins within the same port and with the same peripheral function. These are write-only registers that always read as 0.

75.3.3 EFT detection

The EFT detect circuit is active whenever PORT is powered and EDCR[EDLC] and EDCR[EDHC] are 0.

The EFT high detect circuit for each pin is reset whenever EDCR[EDHC] = 1. Likewise, the EFT low detect circuit for each pin is reset whenever EDCR[EDLC] = 1 and that pin's EFT high detect is not asserted.

75.3.4 Clocking

This module has no clocking considerations.

75.3.5 Interrupt

When EFT on a pin is detected, an interrupt request is generated to indicate the detection of EFT on that pin. This interrupt is enabled when EDIER[EDIE*n*] = 1. Writing 0 to these fields clears the interrupt request but does not affect EFT detection on the corresponding pin (EDFR[EDF*n*]).

75.3.6 Calibration

The combination of Calibration 0 (CALIB0) and Calibration 1 (CALIB1) controls the drive strength of a pin for the port and PCR*n*[DSE] for the pin.

Calibration 0 (CALIB0) and Calibration 1 (CALIB1) represent two driver configurations. If PCR*n*[DSE] = 0, the configuration in Calibration 0 (CALIB0) is used. Likewise, if PCR*n*[DSE] = 1, the configuration in Calibration 1 (CALIB1) is used.

The pulldown and pullup drivers have eight segments. The value of NCAL[5:3] represents the number of pulldown segments that are enabled when driving low, and the value of PCAL[5:3] represents the number of pullup segments that turn on when driving high. Writing 000b to these fields causes one segment to turn on, while writing 111b causes all eight segments to turn on. Upon reset, Calibration 0 (CALIB0) becomes 1 to create approximately 50 Ω driver at 3.3 V and 25 °C (3 is a typical value) and Calibration 1 (CALIB1) becomes 1 to create approximately 50 Ω driver at 1.8 V and 25 °C (7 is the typical value).

The three LSBs of CALIB0[NCAL], CALIB1[NCAL], CALIB0[PCAL], and CALIB1[PCAL] do not affect the driver's strength. Instead, they are included to improve precision when calibrating to a different impedance value. This can be done by using the following formula:

Calibration = (G × (CAL0 + 4) – 1024) >> 8, where G is a binary code representing the desired conductance and CAL0 is one of the 6-bit fields in Calibration 0 (CALIB0) and Calibration 1 (CALIB1). The value of G scales linearly with the intended conductance; G = 0 corresponds to 0 S (an open circuit), and G = 256 corresponds to 0.02 S (50 Ω). For example, the following operation configures the driver to be approximately 0.03 S (33 Ω) at 3.3 V when PCR*n*[DSE] = 1:

The value of CALIB1[NCAL] = (384 × (the value of CALIB0[NCAL] + 4) – 1024) >> 8;

The value of CALIB1[PCAL] = (384 × (the value of CALIB0[PCAL] + 4) – 1024) >> 8;

Do not overflow or underflow NCAL[5:3] or PCAL[5:3]; otherwise, this operation may have an unintended result. You can configure the drive strength for 50 Ω at 2.5 V and 25 °C by averaging the initial NCAL codes in Calibration 0 (CALIB0) and Calibration 1 (CALIB1), and averaging the initial PCAL codes in Calibration 0 (CALIB0) and Calibration 1 (CALIB1). However, the 2.5 V configuration is not as accurate or precise as the original 1.8 V and 3.3 V configuration because the drive strength is not perfectly linear with supply voltage.

NOTE

This calibration feature may not apply to all PORT pins. See the chip-specific PORT information for pins that support calibration.

75.4 Initialization

To initialize PORT, perform the following procedure:

1. Initialize the pin functions:
   - Initialize single pin functions by writing appropriate values to PCR*n*.
   - Initialize multiple pins (up to 16) with the same configuration by writing appropriate values to Global Pin Control Low (GPCLR) or Global Pin Control High (GPCHR).
2. Lock the configuration for a given pin, by writing 1 to PCR*n*[LK], so that it cannot be changed until the next reset.

75.5 Application information

75.5.1 Determine polarity of an asserted EFT detection

You can determine the polarity of an asserted EFT detect by performing this procedure:

1. Read EFT Detect Flag (EDFR).
2. Write 1 and then write 0 to EDCR[EDLC].
3. Read EFT Detect Flag (EDFR) again. Any field that has now become 0 indicates triggering of the EFT low detector only (no EFT high detector triggering).
4. Write 1 and then write 0 to EDCR[EDHC].
5. Read EFT Detect Flag (EDFR) again. Any field that has now become 0 indicates triggering of the EFT high detector. Any field that did not become 0 indicates triggering of both the EFT high and EFT low detectors.

6. Write 1 and then write 0 to both EDCR[EDLC] and EDCR[EDHC] to ensure that all detectors are reset.

75.6 Memory map and register definition

Any read or write access to the PORT memory space, outside the valid memory map, results in a bus error. All register accesses complete with zero wait states.

75.6.1 PORT register descriptions

75.6.1.1 PORT memory map

PORT0 base address: 4011_6000h

PORT1 base address: 4011_7000h

PORT2 base address: 4011_8000h

PORT3 base address: 4011_9000h

PORT4 base address: 4011_A000h

PORT5 base address: 4004_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	R	0200_0000h
10h	Global Pin Control Low (GPCLR)	32	RW	0000_0000h
14h	Global Pin Control High (GPCHR)	32	RW	0000_0000h
20h	Configuration (CONFIG)	32	RW	0000_0000h
40h	EFT Detect Flag (EDFR)	32	R	0000_0000h
44h	EFT Detect Interrupt Enable (EDIER)	32	RW	0000_0000h
48h	EFT Detect Clear (EDCR)	32	RW	0000_0000h
60h	Calibration 0 (CALIB0)	32	RW	See section
64h	Calibration 1 (CALIB1)	32	RW	See section
80h	Pin Control 0 (PCR0)	32	RW	See section
84h	Pin Control 1 (PCR1)	32	RW	See section
88h	Pin Control 2 (PCR2)	32	RW	See section
8Ch	Pin Control 3 (PCR3)	32	RW	See section
90h - 94h	Pin Control a (PCR4 - PCR5)	32	RW	0000_0000h
98h	Pin Control 6 (PCR6)	32	RW	See section
9Ch - A4h	Pin Control a (PCR7 - PCR9)	32	RW	See section
A8h - BCh	Pin Control a (PCR10 - PCR15)	32	RW	See section
C0h	Pin Control 16 (PCR16)	32	RW	0000_0000h
C4h	Pin Control 17 (PCR17)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
C8h - CCh	Pin Control a (PCR18 - PCR19)	32	RW	0000_0000h
D0h	Pin Control 20 (PCR20)	32	RW	0000_0000h
D4h	Pin Control 21 (PCR21)	32	RW	0000_0000h
D8h - FCh	Pin Control a (PCR22 - PCR31)	32	RW	See section

75.6.1.2 Version ID (VERID)

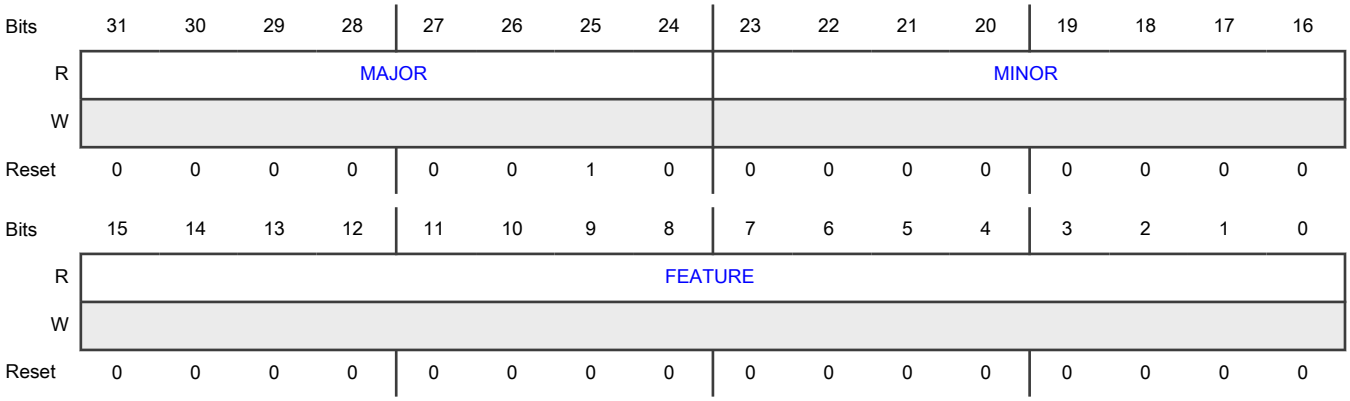
Offset

Register	Offset
VERID	0h

Function

Specifies the version number and feature number of the chip.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Indicates the major version number for the specification.
23-16 MINOR	Minor Version Number Indicates the minor version number for the specification.
15-0	Feature Specification Number

Table continues on the next page...

Table continued from the previous page...

Field	Function
FEATURE	Indicates the feature set number. 0000_0000_0000_0000b - Basic implementation

75.6.1.3 Global Pin Control Low (GPCLR)

Offset

Register	Offset
GPCLR	10h

Function

Controls writes to the PCR15–PCR0 registers.

NOTE

This register supports only 32-bit writes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	GPWE 15	GPWE 14	GPWE 13	GPWE 12	GPWE 11	GPWE 10	GPWE 9	GPWE 8	GPWE 7	GPWE 6	GPWE 5	GPWE 4	GPWE 3	GPWE 2	GPWE 1	GPWE 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	GPWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 GPWEn	Global Pin Write Enable Configures the corresponding lower 16-bit field of PCR*n* to be updated with the value in the GPWD field. If a selected PCR is locked, the write to that register is ignored. 0b - Not updated 1b - Updated
15-0 GPWD	Global Pin Write Data Is written to PCR*n*[15:0] if GPWE*n* = 1.

75.6.1.4 Global Pin Control High (GPCHR)

Offset

Register	Offset
GPCHR	14h

Function

Controls writes to the PCR31–PCR16 registers.

NOTE

This register supports only 32-bit writes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	GPWE 31	GPWE 30	GPWE 29	GPWE 28	GPWE 27	GPWE 26	GPWE 25	GPWE 24	GPWE 23	GPWE 22	GPWE 21	GPWE 20	GPWE 19	GPWE 18	GPWE 17	GPWE 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	GPWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 GPWEn	Global Pin Write Enable Configures the corresponding lower 16-bit field of PCR*n* to be updated with the value in the GPWD field. If a selected PCR is locked, write to that register is ignored. 0b - Not updated 1b - Updated
15-0 GPWD	Global Pin Write Data Is written to PCR*n*[15:0] if GPWE*n* = 1.

75.6.1.5 Configuration (CONFIG)

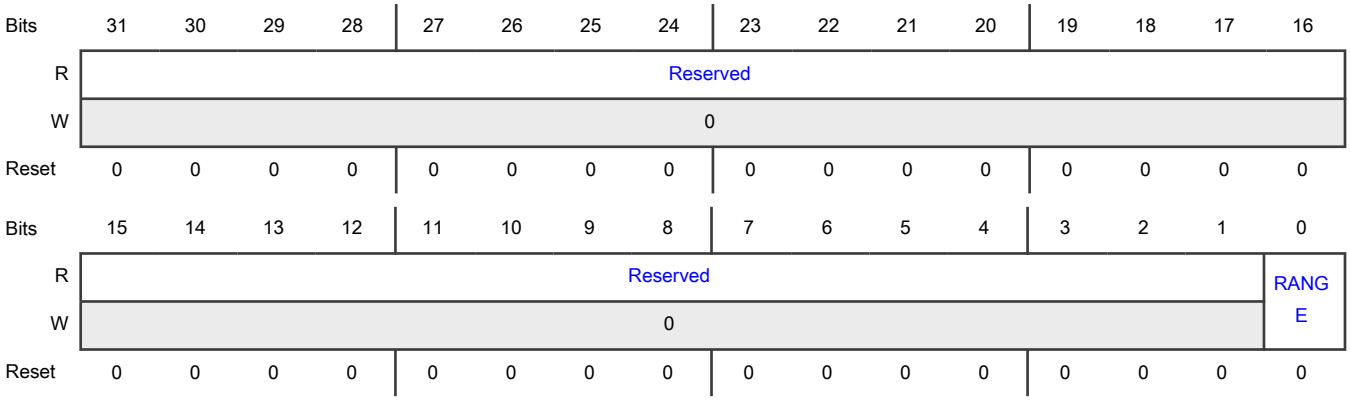
Offset

Register	Offset
CONFIG	20h

Function

Configures the port voltage range.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 RANGE	Port Voltage Range Configures the port voltage range. 0b - 1.71 V–3.6 V 1b - 2.70 V–3.6 V

75.6.1.6 EFT Detect Flag (EDFR)

Offset

Register	Offset
EDFR	40h

Function

Specifies whether an EFT event is detected. The EFT detect logic is active in all pin multiplexing modes.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EDF31	EDF30	EDF29	EDF28	EDF27	EDF26	EDF25	EDF24	EDF23	EDF22	EDF21	EDF20	EDF19	EDF18	EDF17	EDF16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EDF15	EDF14	EDF13	EDF12	EDF11	EDF10	EDF9	EDF8	EDF7	EDF6	EDF5	EDF4	EDF3	EDF2	EDF1	EDF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31
EDF31</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>EDFR</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>—</td><td>EDFR</td></tr><tr><td>PORT4</td><td>—</td><td>EDFR</td></tr><tr><td>PORT5</td><td>—</td><td>EDFR</td></tr></table>0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>30
EDF30</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected
</td></tr>
<tr><td>29
EDF29</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected
</td></tr>
<tr><td>28
EDF28</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected
</td></tr>
<tr><td>27
EDF27</td><td>
EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>26
EDF26</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>EDFR</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>—</td><td>EDFR</td></tr><tr><td>PORT4</td><td>—</td><td>EDFR</td></tr><tr><td>PORT5</td><td>—</td><td>EDFR</td></tr></table>0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>25
EDF25</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>EDFR</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>—</td><td>EDFR</td></tr><tr><td>PORT4</td><td>—</td><td>EDFR</td></tr><tr><td>PORT5</td><td>—</td><td>EDFR</td></tr></table></td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>24
EDF24</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>EDFR</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>—</td><td>EDFR</td></tr><tr><td>PORT4</td><td>—</td><td>EDFR</td></tr><tr><td>PORT5</td><td>—</td><td>EDFR</td></tr></table>0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>23
EDF23</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>EDFR</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>EDFR</td><td>—</td></tr><tr><td>PORT4</td><td>EDFR</td><td>—</td></tr></table></td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Instance / **Field supported in** / **Field not supported in** PORT5 / — / EDFR 0b - No EFT event detected 1b - High or/and low EFT event detected
22 EDF22	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDFR / — PORT1 / EDFR / — PORT2 / — / EDFR PORT3 / EDFR / — PORT4 / EDFR / — PORT5 / — / EDFR 0b - No EFT event detected 1b - High or/and low EFT event detected
21 EDF21	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDFR / — PORT1 / EDFR / —

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>20
EDF20</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>19
EDF19</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>18
EDF18</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>17
EDF17</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected
</td></tr>
<tr><td>16
EDF16</td><td>
EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>15
EDF15</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>EDFR</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>EDFR</td><td>—</td></tr><tr><td>PORT4</td><td>EDFR</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDFR</td></tr></table>
0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>14
EDF14</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>EDFR</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>EDFR</td><td>—</td></tr><tr><td>PORT4</td><td>EDFR</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDFR</td></tr></table></td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>13
EDF13</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>EDFR</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>EDFR</td><td>—</td></tr><tr><td>PORT4</td><td>EDFR</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDFR</td></tr></table>0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>12
EDF12</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>EDFR</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDFR</td></tr><tr><td>PORT3</td><td>EDFR</td><td>—</td></tr><tr><td>PORT4</td><td>EDFR</td><td>—</td></tr></table></td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Instance / **Field supported in** / **Field not supported in** PORT5 / — / EDFR 0b - No EFT event detected 1b - High or/and low EFT event detected
11 EDF11	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDFR / — PORT1 / EDFR / — PORT2 / EDFR / — PORT3 / EDFR / — PORT4 / — / EDFR PORT5 / — / EDFR 0b - No EFT event detected 1b - High or/and low EFT event detected
10 EDF10	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDFR / — PORT1 / EDFR / —

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT2</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDFR</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>9
EDF9</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT2</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT3</td><td>EDFR</td><td>—</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDFR</td></tr>
<tr><td>PORT5</td><td>EDFR</td><td>—</td></tr>
</table>
0b - No EFT event detected
1b - High or/and low EFT event detected</td></tr>
<tr><td>8
EDF8</td><td>EFT Detect Flag
Indicates whether high or low EFT is detected on the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDFR</td><td>—</td></tr><tr><td>PORT1</td><td>EDFR</td><td>—</td></tr><tr><td>PORT2</td><td>EDFR</td><td>—</td></tr><tr><td>PORT3</td><td>EDFR</td><td>—</td></tr><tr><td>PORT4</td><td>—</td><td>EDFR</td></tr><tr><td>PORT5</td><td>EDFR</td><td>—</td></tr></table> 0b - No EFT event detected 1b - High or/and low EFT event detected
7 EDF7	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected 1b - High or/and low EFT event detected
6 EDF6	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected 1b - High or/and low EFT event detected
5 EDF5	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected 1b - High or/and low EFT event detected
4 EDF4	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected 1b - High or/and low EFT event detected
3 EDF3	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - High or/and low EFT event detected
2 EDF2	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected 1b - High or/and low EFT event detected
1 EDF1	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected 1b - High or/and low EFT event detected
0 EDF0	EFT Detect Flag Indicates whether high or low EFT is detected on the corresponding pin. 0b - No EFT event detected 1b - High or/and low EFT event detected

75.6.1.7 EFT Detect Interrupt Enable (EDIER)

Offset

Register	Offset
EDIER	44h

Function

Configures whether to generate an interrupt when an EFT event is detected.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EDIE31	EDIE30	EDIE29	EDIE28	EDIE27	EDIE26	EDIE25	EDIE24	EDIE23	EDIE22	EDIE21	EDIE20	EDIE19	EDIE18	EDIE17	EDIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EDIE15	EDIE14	EDIE13	EDIE12	EDIE11	EDIE10	EDIE9	EDIE8	EDIE7	EDIE6	EDIE5	EDIE4	EDIE3	EDIE2	EDIE1	EDIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31
EDIE31</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>—</td><td>EDIER</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>30
EDIE30</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>—</td><td>EDIER</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table>
0b - Interrupt not generated upon detection of the EFT event</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>29
EDIE29</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>EDIER</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>—</td><td>EDIER</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table>0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>28
EDIE28</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>EDIER</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>—</td><td>EDIER</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr></table></td></tr>
</table>

Table continued from the previous page...

Field	Function
	Instance / **Field supported in** / **Field not supported in** PORT5 / — / EDIER 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
27 EDIE27	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDIER / — PORT1 / — / EDIER PORT2 / — / EDIER PORT3 / — / EDIER PORT4 / — / EDIER PORT5 / — / EDIER 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
26 EDIE26	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDIER / — PORT1 / — / EDIER

Table continues on the next page...

Table continued from the previous page...

Field	Function
	<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>—</td><td>EDIER</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table> 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
25 EDIE25	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>EDIER</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>—</td><td>EDIER</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table> 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
24 EDIE24	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT1</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT3</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>23
EDIE23</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT3</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>22
EDIE22</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT3</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event
</td></tr>
<tr><td>21
EDIE21</td><td>
EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT3</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>20
EDIE20</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>EDIER</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table>0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>19
EDIE19</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>EDIER</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table></td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>18
EDIE18</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>EDIER</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table>0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>17
EDIE17</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>EDIER</td><td>—</td></tr></table></td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Instance / **Field supported in** / **Field not supported in** PORT5 / — / EDIER 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
16 EDIE16	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDIER / — PORT1 / EDIER / — PORT2 / — / EDIER PORT3 / EDIER / — PORT4 / EDIER / — PORT5 / — / EDIER 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
15 EDIE15	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / EDIER / — PORT1 / EDIER / —

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT2</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT3</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>14
EDIE14</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT2</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT3</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT4</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>13
EDIE13</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>EDIER</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table> 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
12 EDIE12	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. **NOTE** This field is not supported in every instance. The following table includes only supported registers. <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>—</td><td>EDIER</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>EDIER</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>EDIER</td></tr></table> 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
11 EDIE11	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin.

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT2</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT3</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event
</td></tr>
<tr><td>10
EDIE10</td><td>
EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT1</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT2</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT3</td><td>EDIER</td><td>—</td></tr>
<tr><td>PORT4</td><td>—</td><td>EDIER</td></tr>
<tr><td>PORT5</td><td>—</td><td>EDIER</td></tr>
</table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>9
EDIE9</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>EDIER</td><td>—</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr><tr><td>PORT5</td><td>EDIER</td><td>—</td></tr></table>
0b - Interrupt not generated upon detection of the EFT event
1b - Interrupt generated upon detection of the EFT event</td></tr>
<tr><td>8
EDIE8</td><td>EFT Detect Interrupt Enable
Configures the EFT detect interrupt for the corresponding pin.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>EDIER</td><td>—</td></tr><tr><td>PORT1</td><td>EDIER</td><td>—</td></tr><tr><td>PORT2</td><td>EDIER</td><td>—</td></tr><tr><td>PORT3</td><td>EDIER</td><td>—</td></tr><tr><td>PORT4</td><td>—</td><td>EDIER</td></tr><tr><td>PORT5</td><td>EDIER</td><td>—</td></tr></table></td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
7 EDIE7	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
6 EDIE6	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
5 EDIE5	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
4 EDIE4	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
3 EDIE3	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
2 EDIE2	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
1 EDIE1	EFT Detect Interrupt Enable Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event
0	EFT Detect Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
EDIE0	Configures the EFT detect interrupt for the corresponding pin. 0b - Interrupt not generated upon detection of the EFT event 1b - Interrupt generated upon detection of the EFT event

75.6.1.8 EFT Detect Clear (EDCR)

Offset

Register	Offset
EDCR	48h

Function

Clears EFT detectors. The EFT detect logic is cleared per port.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0														EDLC	EDHC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-2 —	Reserved
1 EDLC	EFT Detect Low Clear Clears low EFT detectors. If this field = 1, all low EFT detectors for which the corresponding high EFT detectors are not asserted are cleared. 0b - Does not clear 1b - Clears
0	EFT Detect High Clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
EDHC	Clears high EFT detectors. 0b - Does not clear 1b - Clears

75.6.1.9 Calibration 0 (CALIB0)

Offset

Register	Offset
CALIB0	60h

Function

Stores calibration values for the PMOS and NMOS output drivers when PCR*n*[DSE] = 0.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	CALIB0	—
PORT1	CALIB0	—
PORT2	CALIB0	—
PORT3	CALIB0	—
PORT4	—	CALIB0
PORT5	—	CALIB0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										PCAL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	u[1]	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										NCAL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	u[1]	u	u	u	u	u

1. Reset values are loaded out of IFR.

Fields

Field	Function
31-22 —	Reserved
21-16 PCAL	Calibration of PMOS Output Driver
15-6 —	Reserved
5-0 NCAL	Calibration of NMOS Output Driver

75.6.1.10 Calibration 1 (CALIB1)

Offset

Register	Offset
CALIB1	64h

Function
Stores calibration values for the PMOS and NMOS output drivers when PCR*n*[DSE] = 1.

NOTE
Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	CALIB1	—
PORT1	CALIB1	—
PORT2	CALIB1	—
PORT3	CALIB1	—
PORT4	—	CALIB1
PORT5	—	CALIB1

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										PCAL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	u[1]	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										NCAL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	u[1]	u	u	u	u	u

1. Reset values are loaded out of IFR.

Fields

Field	Function
31-22 —	Reserved
21-16 PCAL	Calibration of PMOS Output Driver
15-6 —	Reserved
5-0 NCAL	Calibration of NMOS Output Driver

75.6.1.11 Pin Control 0 (PCR0)

Offset

Register	Offset
PCR0	80h

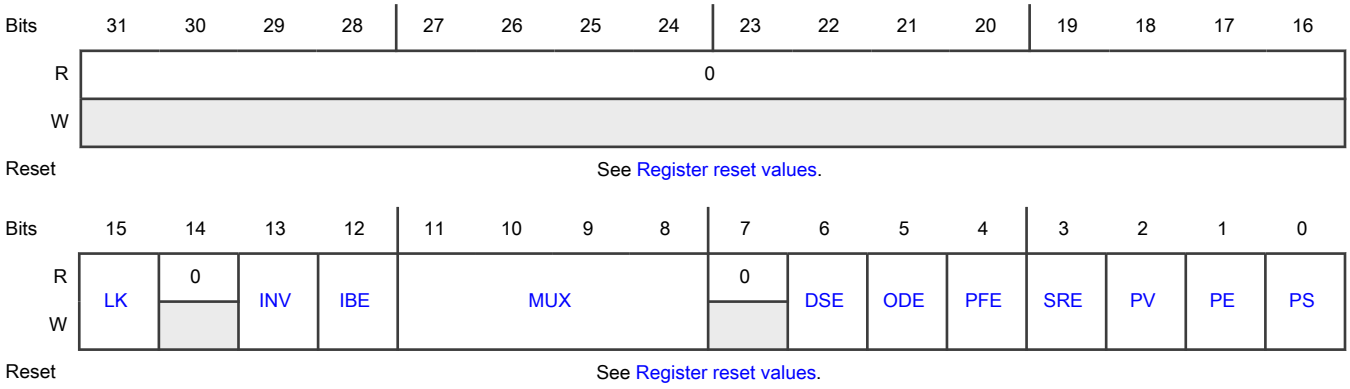
Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

Diagram



Register reset values

Register	Reset value
PCR0	PORT0: 0000_1143h PORT1–PORT5: 0000_0000h

Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Enables</td></tr>
<tr><td>11-8
MUX</td><td>Pin Multiplex Control
Configures the multiplexing slots on each pin.
Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved.
Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die.
The corresponding pin is configured according to the following pin multiplexing slots:
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT5</td><td>PCR0[9–8]</td><td>PCR0[11–10]</td></tr>
</table>
NOTE
The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT0
PORT1
PORT2
PORT3
PORT4</td><td>0000b - Alternative 0 (GPIO)
0001b - Alternative 1 (chip-specific)
0010b - Alternative 2 (chip-specific)
0011b - Alternative 3 (chip-specific)
0100b - Alternative 4 (chip-specific)
0101b - Alternative 5 (chip-specific)
0110b - Alternative 6 (chip-specific)
0111b - Alternative 7 (chip-specific)</td></tr>
</table>
</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td></td><td>1000b - Alternative 8 (chip-specific)
1001b - Alternative 9 (chip-specific)
1010b - Alternative 10 (chip-specific)
1011b - Alternative 11 (chip-specific)
1100b - Alternative 12 (chip-specific)
1101b - Alternative 13 (chip-specific)</td></tr>
<tr><td>PORT5</td><td>00b - Alternative 0 (GPIO)
01b - Alternative 1 (chip-specific)
10b - Alternative 2 (chip-specific)
11b - Alternative 3 (chip-specific)</td></tr>
</table>
</td></tr>
<tr><td>7
—</td><td>Reserved</td></tr>
<tr><td>6
DSE</td><td>Drive Strength Enable
Configures drive strength, low or high, on each pin.
The drive strength configuration is valid for all digital pin multiplexing modes.
• When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
• When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR0</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR0</td><td>—</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Instance / **Field supported in** / **Field not supported in** PORT5 / — / PCR0 0b - Low 1b - High
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables 1b - Enables
4 PFE	Passive Filter Enable Enables passive input filter on each pin. The passive filter configuration is valid for all digital pin multiplexing modes. • When this field = 0, the passive input filter is disabled on the corresponding pin. • When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input. See the chip's data sheet for filter characteristics. **NOTE** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** PORT0 / — / PCR0 PORT1 / PCR0 / — PORT2 / — / PCR0 PORT3 / — / PCR0 PORT4 / — / PCR0 PORT5 / PCR0 / —

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Instance / **Field supported in** / **Field not supported in** 0b - Disables 1b - Enables
3 SRE	Slew Rate Enable Configures the slew rate feature, fast or slow, on each corresponding pin. The slew rate configuration is valid for all digital pin multiplexing modes. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PORT0	PCR0	—
PORT1	PCR0	—
PORT2	PCR0	—
PORT3	PCR0	—
PORT4	PCR0	—
PORT5	—	PCR0

0b - Fast

1b - Slow

Field	Function
2 PV	Pull Value Selects high or low internal pull resistor value. The pull value configuration is valid for all digital pin multiplexing modes. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PORT0	—	PCR0
PORT1	—	PCR0

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT2</td><td>—</td><td>PCR0</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR0</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR0</td></tr>
<tr><td>PORT5</td><td>PCR0</td><td>—</td></tr>
</table>
0b - Low
1b - High</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
0b - Enables internal pulldown resistor
1b - Enables internal pullup resistor</td></tr>
</table>

75.6.1.12 Pin Control 1 (PCR1)

Offset

Register	Offset
PCR1	84h

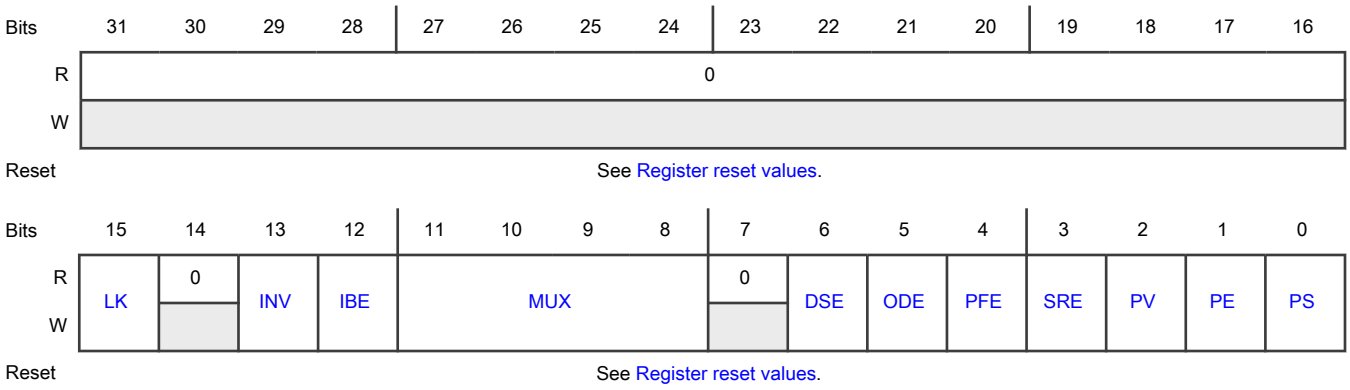
Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

Diagram



Register reset values

Register	Reset value
PCR1	PORT0: 0000_1102h PORT1–PORT5: 0000_0000h

Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input.

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Does not invert
1b - Inverts</td></tr>
<tr><td>12
IBE</td><td>Input Buffer Enable
Enables digital input buffer. When disabled, the digital input is required for analog functions.
0b - Disables
1b - Enables</td></tr>
<tr><td>11-8
MUX</td><td>Pin Multiplex Control
Configures the multiplexing slots on each pin.
Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved.
Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die.
The corresponding pin is configured according to the following pin multiplexing slots:
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT5</td><td>PCR1[9–8]</td><td>PCR1[11–10]</td></tr>
</table>
NOTE
The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT0
PORT1
PORT2
PORT3
PORT4</td><td>0000b - Alternative 0 (GPIO)
0001b - Alternative 1 (chip-specific)
0010b - Alternative 2 (chip-specific)</td></tr>
</table>
</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td></td><td>0011b - Alternative 3 (chip-specific)
0100b - Alternative 4 (chip-specific)
0101b - Alternative 5 (chip-specific)
0110b - Alternative 6 (chip-specific)
0111b - Alternative 7 (chip-specific)
1000b - Alternative 8 (chip-specific)
1001b - Alternative 9 (chip-specific)
1010b - Alternative 10 (chip-specific)
1011b - Alternative 11 (chip-specific)
1100b - Alternative 12 (chip-specific)
1101b - Alternative 13 (chip-specific)</td></tr>
<tr><td>PORT5</td><td>00b - Alternative 0 (GPIO)
01b - Alternative 1 (chip-specific)
10b - Alternative 2 (chip-specific)
11b - Alternative 3 (chip-specific)</td></tr>
</table>
</td></tr>
<tr><td>7
—</td><td>Reserved</td></tr>
<tr><td>6
DSE</td><td>Drive Strength Enable
Configures drive strength, low or high, on each pin.
The drive strength configuration is valid for all digital pin multiplexing modes.
• When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
• When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR1</td><td>—</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT1</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR1</td></tr>
</table>
0b - Low
1b - High
</td></tr>
<tr><td>5
ODE</td><td>Open Drain Enable
Enables open drain output on each pin.
The open drain configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the open drain output is disabled on the corresponding pin.
• When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
0b - Disables
1b - Enables</td></tr>
<tr><td>4
PFE</td><td>Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the passive input filter is disabled on the corresponding pin.
• When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.
See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT1</td><td>PCR1</td><td>—</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT2</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT5</td><td>PCR1</td><td>—</td></tr>
</table>
0b - Disables
1b - Enables
</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR1</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR1</td></tr>
</table>
0b - Fast
1b - Slow
</td></tr>
<tr><td>2
PV</td><td>Pull Value
Selects high or low internal pull resistor value.
The pull value configuration is valid for all digital pin multiplexing modes.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT1</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR1</td></tr>
<tr><td>PORT5</td><td>PCR1</td><td>—</td></tr>
</table>
0b - Low
1b - High
</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
0b - Enables internal pulldown resistor
1b - Enables internal pullup resistor</td></tr>
</table>

75.6.1.13 Pin Control 2 (PCR2)

Offset

Register	Offset
PCR2	88h

Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	See Register reset values.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0	INV	IBE	MUX				0	DSE	ODE	PFE	SRE	PV	PE	PS
W																
Reset	See Register reset values.															

Register reset values

Register	Reset value
PCR2	PORT0: 0000_0140h PORT1–PORT4: 0000_0000h PORT5: 0000_0100h

Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots: **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PORT0	PCR2	—
PORT1	PCR2	—
PORT2	PCR2	—
PORT3	PCR2	—
PORT4	PCR2	—
PORT5	PCR2[9–8]	PCR2[11–10]

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT0
PORT1
PORT2
PORT3
PORT4</td><td>0000b - Alternative 0 (GPIO)
0001b - Alternative 1 (chip-specific)
0010b - Alternative 2 (chip-specific)
0011b - Alternative 3 (chip-specific)
0100b - Alternative 4 (chip-specific)
0101b - Alternative 5 (chip-specific)
0110b - Alternative 6 (chip-specific)
0111b - Alternative 7 (chip-specific)
1000b - Alternative 8 (chip-specific)
1001b - Alternative 9 (chip-specific)
1010b - Alternative 10 (chip-specific)
1011b - Alternative 11 (chip-specific)
1100b - Alternative 12 (chip-specific)
1101b - Alternative 13 (chip-specific)</td></tr>
<tr><td>PORT5</td><td>00b - Alternative 0 (GPIO)
01b - Alternative 1 (chip-specific)
10b - Alternative 2 (chip-specific)
11b - Alternative 3 (chip-specific)</td></tr>
</table>
</td></tr>
<tr><td>7
—</td><td>Reserved</td></tr>
<tr><td>6
DSE</td><td>Drive Strength Enable
Configures drive strength, low or high, on each pin.
The drive strength configuration is valid for all digital pin multiplexing modes.
• When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
• When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR2</td></tr>
</table>
0b - Low
1b - High
</td></tr>
<tr><td>5
ODE</td><td>Open Drain Enable
Enables open drain output on each pin.
The open drain configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the open drain output is disabled on the corresponding pin.
• When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
0b - Disables
1b - Enables</td></tr>
<tr><td>4
PFE</td><td>Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the passive input filter is disabled on the corresponding pin.
• When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.
See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT1</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT5</td><td>PCR2</td><td>—</td></tr>
</table>
0b - Disables
1b - Enables
</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR2</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR2</td></tr>
</table>
0b - Fast
1b - Slow
</td></tr>
<tr><td>2
PV</td><td>Pull Value
Selects high or low internal pull resistor value.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>The pull value configuration is valid for all digital pin multiplexing modes.

NOTE
This field is not supported in every instance. The following table includes only supported registers.

<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT1</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR2</td></tr>
<tr><td>PORT5</td><td>PCR2</td><td>—</td></tr>
</table>
0b - Low
1b - High</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
0b - Enables internal pulldown resistor
1b - Enables internal pullup resistor</td></tr>
</table>

75.6.1.14 Pin Control 3 (PCR3)

Offset

Register	Offset
PCR3	8Ch

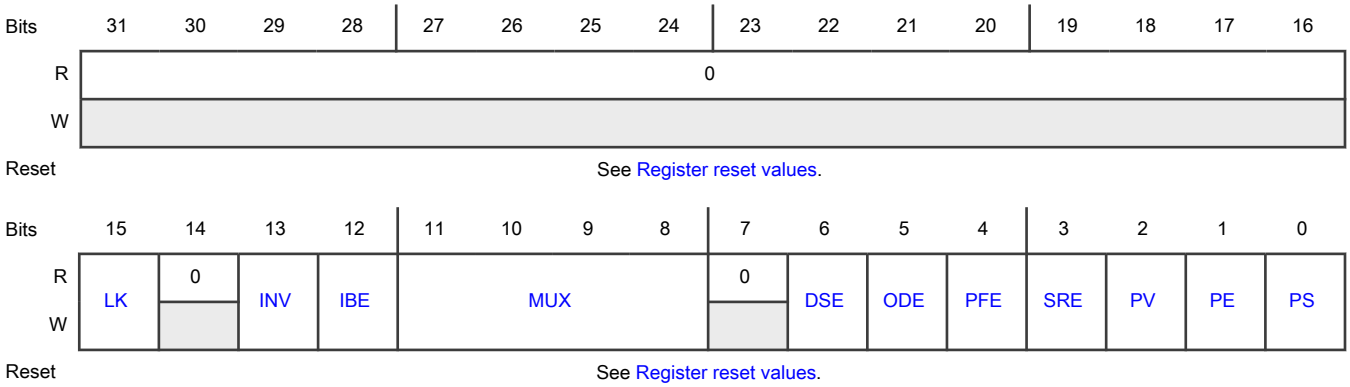
Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

Diagram



Register reset values

Register	Reset value
PCR3	PORT0: 0000_1103h PORT1–PORT5: 0000_0000h

Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots: **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PORT0	PCR3	—
PORT1	PCR3	—
PORT2	PCR3	—
PORT3	PCR3	—
PORT4	PCR3	—
PORT5	PCR3[9–8]	PCR3[11–10]

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT0
PORT1
PORT2
PORT3
PORT4</td><td>0000b - Alternative 0 (GPIO)
0001b - Alternative 1 (chip-specific)
0010b - Alternative 2 (chip-specific)
0011b - Alternative 3 (chip-specific)
0100b - Alternative 4 (chip-specific)
0101b - Alternative 5 (chip-specific)
0110b - Alternative 6 (chip-specific)
0111b - Alternative 7 (chip-specific)
1000b - Alternative 8 (chip-specific)
1001b - Alternative 9 (chip-specific)
1010b - Alternative 10 (chip-specific)
1011b - Alternative 11 (chip-specific)
1100b - Alternative 12 (chip-specific)
1101b - Alternative 13 (chip-specific)</td></tr>
<tr><td>PORT5</td><td>00b - Alternative 0 (GPIO)
01b - Alternative 1 (chip-specific)
10b - Alternative 2 (chip-specific)
11b - Alternative 3 (chip-specific)</td></tr>
</table>
</td></tr>
<tr><td>7
—</td><td>Reserved</td></tr>
<tr><td>6
DSE</td><td>Drive Strength Enable
Configures drive strength, low or high, on each pin.
The drive strength configuration is valid for all digital pin multiplexing modes.
• When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
• When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR3</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR3</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR3</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR3</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR3</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR3</td></tr>
</table>
0b - Low
1b - High
</td></tr>
<tr><td>5
ODE</td><td>Open Drain Enable
Enables open drain output on each pin.
The open drain configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the open drain output is disabled on the corresponding pin.
• When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
0b - Disables
1b - Enables</td></tr>
<tr><td>4
PFE</td><td>Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the passive input filter is disabled on the corresponding pin.
• When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.
See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function

Instance	Field supported in	Field not supported in
PORT0	—	PCR3
PORT1	—	PCR3
PORT2	—	PCR3
PORT3	—	PCR3
PORT4	—	PCR3
PORT5	PCR3	—

0b - Disables

1b - Enables

Field	Function
3 SRE	Slew Rate Enable Configures the slew rate feature, fast or slow, on each corresponding pin. The slew rate configuration is valid for all digital pin multiplexing modes. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PORT0	PCR3	—
PORT1	PCR3	—
PORT2	PCR3	—
PORT3	PCR3	—
PORT4	PCR3	—
PORT5	—	PCR3

0b - Fast

1b - Slow

Field	Function
2 PV	Pull Value Selects high or low internal pull resistor value.

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>The pull value configuration is valid for all digital pin multiplexing modes.

NOTE
This field is not supported in every instance. The following table includes only supported registers.

<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR3</td></tr>
<tr><td>PORT1</td><td>—</td><td>PCR3</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR3</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR3</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR3</td></tr>
<tr><td>PORT5</td><td>PCR3</td><td>—</td></tr>
</table>
0b - Low
1b - High</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
0b - Enables internal pulldown resistor
1b - Enables internal pullup resistor</td></tr>
</table>

75.6.1.15 Pin Control a (PCR4 - PCR5)

Offset

Register	Offset
PCR4	90h
PCR5	94h

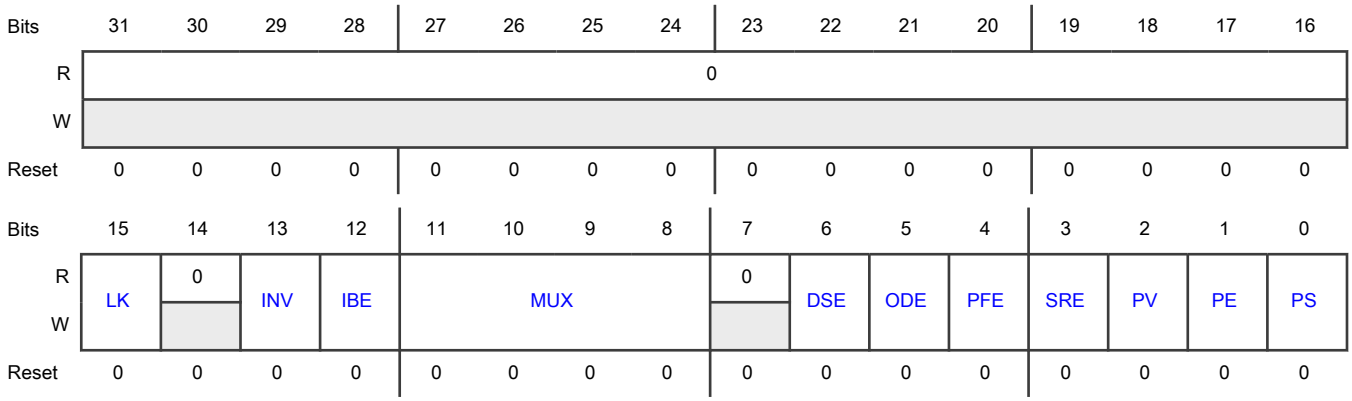
Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

Diagram



Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13	Invert Input

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>INV</td><td>Inverts the digital input.
0b - Does not invert
1b - Inverts</td></tr>
<tr><td>12
IBE</td><td>Input Buffer Enable
Enables digital input buffer. When disabled, the digital input is required for analog functions.
0b - Disables
1b - Enables</td></tr>
<tr><td>11-8
MUX</td><td>Pin Multiplex Control
Configures the multiplexing slots on each pin.
Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved.
Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die.
The corresponding pin is configured according to the following pin multiplexing slots:
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT5</td><td>PCR4–PCR5[9–8]</td><td>PCR4–PCR5[11–10]</td></tr>
</table>
NOTE
The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT0
PORT1
PORT2</td><td>0000b - Alternative 0 (GPIO)
0001b - Alternative 1 (chip-specific)</td></tr>
</table>
</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT3
PORT4</td><td>0010b - Alternative 2 (chip-specific)
0011b - Alternative 3 (chip-specific)
0100b - Alternative 4 (chip-specific)
0101b - Alternative 5 (chip-specific)
0110b - Alternative 6 (chip-specific)
0111b - Alternative 7 (chip-specific)
1000b - Alternative 8 (chip-specific)
1001b - Alternative 9 (chip-specific)
1010b - Alternative 10 (chip-specific)
1011b - Alternative 11 (chip-specific)
1100b - Alternative 12 (chip-specific)
1101b - Alternative 13 (chip-specific)</td></tr>
<tr><td>PORT5</td><td>00b - Alternative 0 (GPIO)
01b - Alternative 1 (chip-specific)
10b - Alternative 2 (chip-specific)
11b - Alternative 3 (chip-specific)</td></tr>
</table>
</td></tr>
<tr><td>7
—</td><td>Reserved</td></tr>
<tr><td>6
DSE</td><td>Drive Strength Enable
Configures drive strength, low or high, on each pin.
The drive strength configuration is valid for all digital pin multiplexing modes.
• When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
• When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>PCR4–PCR5</td><td>—</td></tr><tr><td>PORT1</td><td>PCR4–PCR5</td><td>—</td></tr><tr><td>PORT2</td><td>PCR4–PCR5</td><td>—</td></tr><tr><td>PORT3</td><td>PCR4–PCR5</td><td>—</td></tr><tr><td>PORT4</td><td>PCR4–PCR5</td><td>—</td></tr><tr><td>PORT5</td><td>—</td><td>PCR4–PCR5</td></tr></table> 0b - Low 1b - High
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables 1b - Enables
4 PFE	Passive Filter Enable Enables passive input filter on each pin. The passive filter configuration is valid for all digital pin multiplexing modes. • When this field = 0, the passive input filter is disabled on the corresponding pin. • When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input. See the chip's data sheet for filter characteristics. **NOTE** This field is not supported in every instance. The following table includes only supported registers. <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>PCR4–PCR5</td><td>—</td></tr></table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT1</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT5</td><td>PCR4–PCR5</td><td>—</td></tr>
</table>
0b - Disables
1b - Enables</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR4–PCR5</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR4–PCR5</td></tr>
</table>
0b - Fast
1b - Slow</td></tr>
<tr><td>2
PV</td><td>Pull Value
Selects high or low internal pull resistor value.
The pull value configuration is valid for all digital pin multiplexing modes.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT1</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR4–PCR5</td></tr>
<tr><td>PORT5</td><td>PCR4–PCR5</td><td>—</td></tr>
</table>
0b - Low
1b - High
</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
0b - Enables internal pulldown resistor
1b - Enables internal pullup resistor</td></tr>
</table>

75.6.1.16 Pin Control 6 (PCR6)

Offset

Register	Offset
PCR6	98h

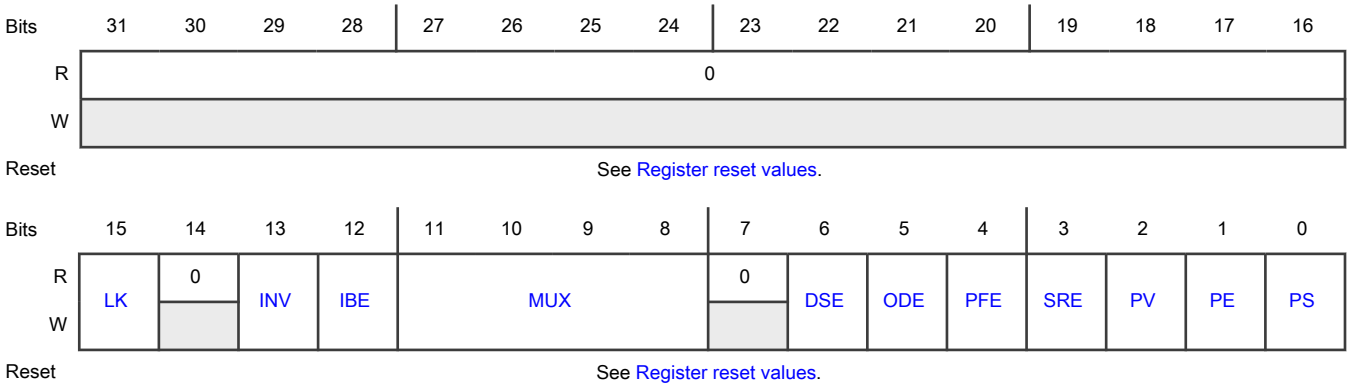
Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

Diagram



Register reset values

Register	Reset value
PCR6	PORT0: 0000_1103h PORT1–PORT5: 0000_0000h

Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots: **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PORT0	PCR6	—
PORT1	PCR6	—
PORT2	PCR6	—
PORT3	PCR6	—
PORT4	PCR6	—
PORT5	PCR6[9–8]	PCR6[11–10]

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT0
PORT1
PORT2
PORT3
PORT4</td><td>0000b - Alternative 0 (GPIO)
0001b - Alternative 1 (chip-specific)
0010b - Alternative 2 (chip-specific)
0011b - Alternative 3 (chip-specific)
0100b - Alternative 4 (chip-specific)
0101b - Alternative 5 (chip-specific)
0110b - Alternative 6 (chip-specific)
0111b - Alternative 7 (chip-specific)
1000b - Alternative 8 (chip-specific)
1001b - Alternative 9 (chip-specific)
1010b - Alternative 10 (chip-specific)
1011b - Alternative 11 (chip-specific)
1100b - Alternative 12 (chip-specific)
1101b - Alternative 13 (chip-specific)</td></tr>
<tr><td>PORT5</td><td>00b - Alternative 0 (GPIO)
01b - Alternative 1 (chip-specific)
10b - Alternative 2 (chip-specific)
11b - Alternative 3 (chip-specific)</td></tr>
</table>
</td></tr>
<tr><td>7
—</td><td>Reserved</td></tr>
<tr><td>6
DSE</td><td>Drive Strength Enable
Configures drive strength, low or high, on each pin.
The drive strength configuration is valid for all digital pin multiplexing modes.
• When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
• When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR6</td></tr>
</table>
0b - Low
1b - High
</td></tr>
<tr><td>5
ODE</td><td>Open Drain Enable
Enables open drain output on each pin.
The open drain configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the open drain output is disabled on the corresponding pin.
• When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
0b - Disables
1b - Enables</td></tr>
<tr><td>4
PFE</td><td>Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the passive input filter is disabled on the corresponding pin.
• When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.
See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT1</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT5</td><td>PCR6</td><td>—</td></tr>
</table>
0b - Disables
1b - Enables
</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR6</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR6</td></tr>
</table>
0b - Fast
1b - Slow
</td></tr>
<tr><td>2
PV</td><td>Pull Value
Selects high or low internal pull resistor value.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>The pull value configuration is valid for all digital pin multiplexing modes.

NOTE
This field is not supported in every instance. The following table includes only supported registers.

<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT1</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR6</td></tr>
<tr><td>PORT5</td><td>PCR6</td><td>—</td></tr>
</table>
0b - Low
1b - High</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
0b - Enables internal pulldown resistor
1b - Enables internal pullup resistor</td></tr>
</table>

75.6.1.17 Pin Control a (PCR7 - PCR9)

Offset

Register	Offset
PCR7	9Ch
PCR8	A0h
PCR9	A4h

Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	PCR7–PCR9	—
PORT1	PCR7–PCR9	—
PORT2	PCR7–PCR9	—
PORT3	PCR7–PCR9	—
PORT4	PCR7	PCR8–PCR9
PORT5	PCR7–PCR9	—

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	See Register reset values.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0	INV	IBE	MUX				0	DSE	ODE	PFE	SRE	PV	PE	PS
W																
Reset	See Register reset values.															

Register reset values

Register	Reset value
PCR7	PORT0–PORT5: 0000_0000h
PCR8–PCR9	PORT0–PORT3: 0000_0000h PORT4: Register not supported PORT5: 0000_0000h

Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots:

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR7</td><td>—</td></tr>
<tr><td>PORT5</td><td>PCR7–PCR9[9–8]</td><td>PCR7–PCR9[11–10]</td></tr>
</table>
NOTE
The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>PORT0
PORT1
PORT2
PORT3
PORT4</td><td>0000b - Alternative 0 (GPIO)
0001b - Alternative 1 (chip-specific)
0010b - Alternative 2 (chip-specific)
0011b - Alternative 3 (chip-specific)
0100b - Alternative 4 (chip-specific)
0101b - Alternative 5 (chip-specific)
0110b - Alternative 6 (chip-specific)
0111b - Alternative 7 (chip-specific)
1000b - Alternative 8 (chip-specific)
1001b - Alternative 9 (chip-specific)
1010b - Alternative 10 (chip-specific)
1011b - Alternative 11 (chip-specific)
1100b - Alternative 12 (chip-specific)
1101b - Alternative 13 (chip-specific)</td></tr>
<tr><td>PORT5</td><td>00b - Alternative 0 (GPIO)</td></tr>
</table>
</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td></td><td>01b - Alternative 1 (chip-specific)
10b - Alternative 2 (chip-specific)
11b - Alternative 3 (chip-specific)</td></tr>
</table>
</td></tr>
<tr><td>7
—</td><td>Reserved</td></tr>
<tr><td>6
DSE</td><td>Drive Strength Enable
Configures drive strength, low or high, on each pin.
The drive strength configuration is valid for all digital pin multiplexing modes.
• When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
• When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR7</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR7–PCR9</td></tr>
</table>
0b - Low
1b - High</td></tr>
<tr><td>5
ODE</td><td>Open Drain Enable
Enables open drain output on each pin.
The open drain configuration is valid for all digital pin multiplexing modes.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>

When this field = 0, the open drain output is disabled on the corresponding pin.
When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
0b - Disables
1b - Enables

</td></tr>
<tr><td>4
PFE</td><td>
Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.

When this field = 0, the passive input filter is disabled on the corresponding pin.
When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.

See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR7–PCR9</td></tr>
<tr><td>PORT1</td><td>PCR8–PCR9</td><td>PCR7</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR7–PCR9</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR7–PCR9</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR7</td></tr>
<tr><td>PORT5</td><td>PCR7–PCR9</td><td>—</td></tr>
</table>
0b - Disables
1b - Enables
</td></tr>
<tr><td>3
SRE</td><td>
Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT1</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT2</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT3</td><td>PCR7–PCR9</td><td>—</td></tr>
<tr><td>PORT4</td><td>PCR7</td><td>—</td></tr>
<tr><td>PORT5</td><td>—</td><td>PCR7–PCR9</td></tr>
</table>
0b - Fast
1b - Slow</td></tr>
<tr><td>2
PV</td><td>Pull Value
Selects high or low internal pull resistor value.
The pull value configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT0</td><td>—</td><td>PCR7–PCR9</td></tr>
<tr><td>PORT1</td><td>PCR8</td><td>PCR7
PCR9</td></tr>
<tr><td>PORT2</td><td>—</td><td>PCR7–PCR9</td></tr>
<tr><td>PORT3</td><td>—</td><td>PCR7–PCR9</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR7</td></tr>
<tr><td>PORT5</td><td>PCR7–PCR9</td><td>—</td></tr>
</table>
0b - Low
1b - High</td></tr>
<tr><td>1</td><td>Pull Enable</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
PE	Enables the internal pull resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pull resistor is not enabled on the corresponding pin. • When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. 0b - Disables 1b - Enables
0 PS	Pull Select Enables the internal pullup or pulldown resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. • When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. 0b - Enables internal pulldown resistor 1b - Enables internal pullup resistor

75.6.1.18 Pin Control a (PCR10 - PCR15)

Offset

Register	Offset
PCR10	A8h
PCR11	ACh
PCR12	B0h
PCR13	B4h
PCR14	B8h
PCR15	BCh

Function

Configures pin control features on each pin.

NOTE

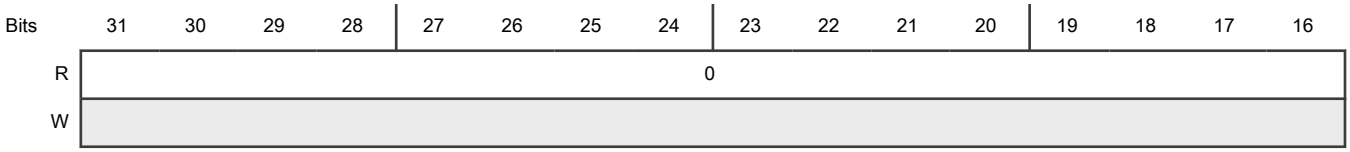
Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

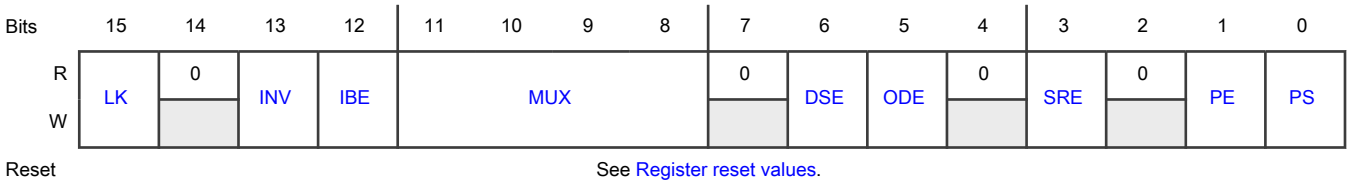
NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	PCR10–PCR15	—
PORT1	PCR10–PCR15	—
PORT2	PCR10–PCR11	PCR12–PCR15
PORT3	PCR10–PCR15	—
PORT4	PCR12–PCR15	PCR10–PCR11
PORT5	—	PCR10–PCR15

Diagram





Register reset values

Register	Reset value
PCR10–PCR11	PORT0–PORT3: 0000_0000h PORT4,PORT5: Register not supported
PCR12–PCR15	PORT0,PORT1: 0000_0000h PORT2: Register not supported PORT3,PORT4: 0000_0000h PORT5: Register not supported

Fields

Field	Function
31-16 —	Reserved
15	Lock Register

Table continues on the next page...

Table continued from the previous page...

Field	Function
LK	Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots: 0000b - Alternative 0 (GPIO) 0001b - Alternative 1 (chip-specific) 0010b - Alternative 2 (chip-specific) 0011b - Alternative 3 (chip-specific) 0100b - Alternative 4 (chip-specific) 0101b - Alternative 5 (chip-specific) 0110b - Alternative 6 (chip-specific) 0111b - Alternative 7 (chip-specific) 1000b - Alternative 8 (chip-specific) 1001b - Alternative 9 (chip-specific) 1010b - Alternative 10 (chip-specific) 1011b - Alternative 11 (chip-specific)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1100b - Alternative 12 (chip-specific) 1101b - Alternative 13 (chip-specific)
7 —	Reserved
6 DSE	Drive Strength Enable Configures drive strength, low or high, on each pin. The drive strength configuration is valid for all digital pin multiplexing modes. • When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output. • When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output. 0b - Low 1b - High
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables 1b - Enables
4 —	Reserved
3 SRE	Slew Rate Enable Configures the slew rate feature, fast or slow, on each corresponding pin. The slew rate configuration is valid for all digital pin multiplexing modes. 0b - Fast 1b - Slow
2 —	Reserved
1 PE	Pull Enable Enables the internal pull resistor.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pull resistor is not enabled on the corresponding pin. • When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. 0b - Disables 1b - Enables
0 PS	Pull Select Enables the internal pullup or pulldown resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. • When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. 0b - Enables internal pulldown resistor 1b - Enables internal pullup resistor

75.6.1.19 Pin Control 16 (PCR16)

Offset

Register	Offset
PCR16	C0h

Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

NOTE

Each module instance supports a different number of registers.

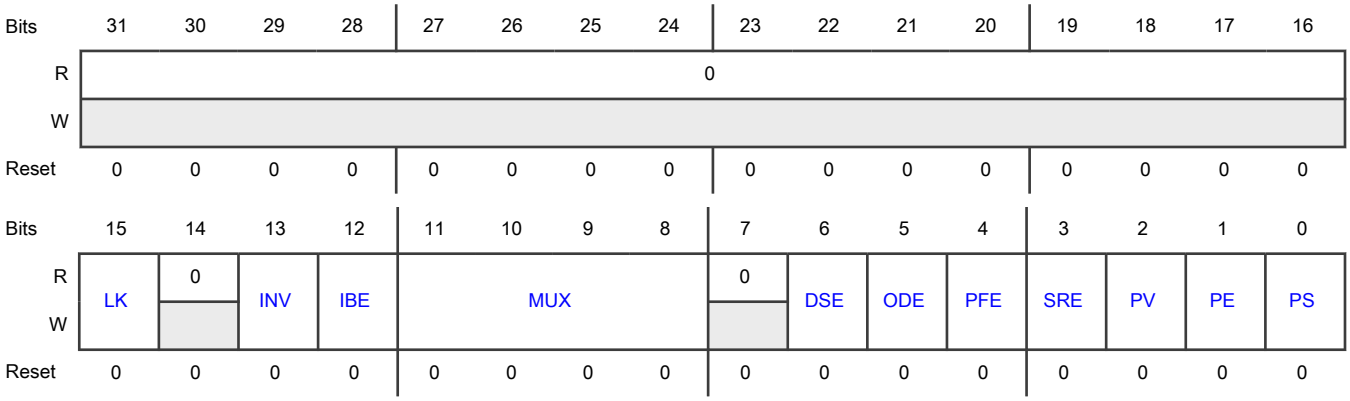
Instance	Register supported	Register not supported
PORT0	PCR16	—
PORT1	PCR16	—

Table continues on the next page...

Table continued from the previous page...

Instance	Register supported	Register not supported
PORT2	—	PCR16
PORT3	PCR16	—
PORT4	PCR16	—
PORT5	—	PCR16

Diagram



Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots: 0000b - Alternative 0 (GPIO) 0001b - Alternative 1 (chip-specific) 0010b - Alternative 2 (chip-specific) 0011b - Alternative 3 (chip-specific) 0100b - Alternative 4 (chip-specific) 0101b - Alternative 5 (chip-specific) 0110b - Alternative 6 (chip-specific) 0111b - Alternative 7 (chip-specific) 1000b - Alternative 8 (chip-specific) 1001b - Alternative 9 (chip-specific) 1010b - Alternative 10 (chip-specific) 1011b - Alternative 11 (chip-specific) 1100b - Alternative 12 (chip-specific) 1101b - Alternative 13 (chip-specific)
7 —	Reserved
6 DSE	Drive Strength Enable Configures drive strength, low or high, on each pin. The drive strength configuration is valid for all digital pin multiplexing modes. • When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output. • When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output.

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Low
1b - High</td></tr>
<tr><td>5
ODE</td><td>Open Drain Enable
Enables open drain output on each pin.
The open drain configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the open drain output is disabled on the corresponding pin.
• When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
0b - Disables
1b - Enables</td></tr>
<tr><td>4
PFE</td><td>Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the passive input filter is disabled on the corresponding pin.
• When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.
See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>PCR16</td><td>—</td></tr><tr><td>PORT1</td><td>PCR16</td><td>—</td></tr><tr><td>PORT3</td><td>—</td><td>PCR16</td></tr><tr><td>PORT4</td><td>—</td><td>PCR16</td></tr></table>0b - Disables
1b - Enables</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Fast
1b - Slow</td></tr>
<tr><td>2
PV</td><td>Pull Value
Selects high or low internal pull resistor value.
The pull value configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>PCR16</td><td>—</td></tr><tr><td>PORT1</td><td>PCR16</td><td>—</td></tr><tr><td>PORT3</td><td>—</td><td>PCR16</td></tr><tr><td>PORT4</td><td>—</td><td>PCR16</td></tr></table>0b - Low
1b - High</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enables internal pulldown resistor 1b - Enables internal pullup resistor

75.6.1.20 Pin Control 17 (PCR17)

Offset

Register	Offset
PCR17	C4h

Function

Configures pin control features on each pin.

NOTE

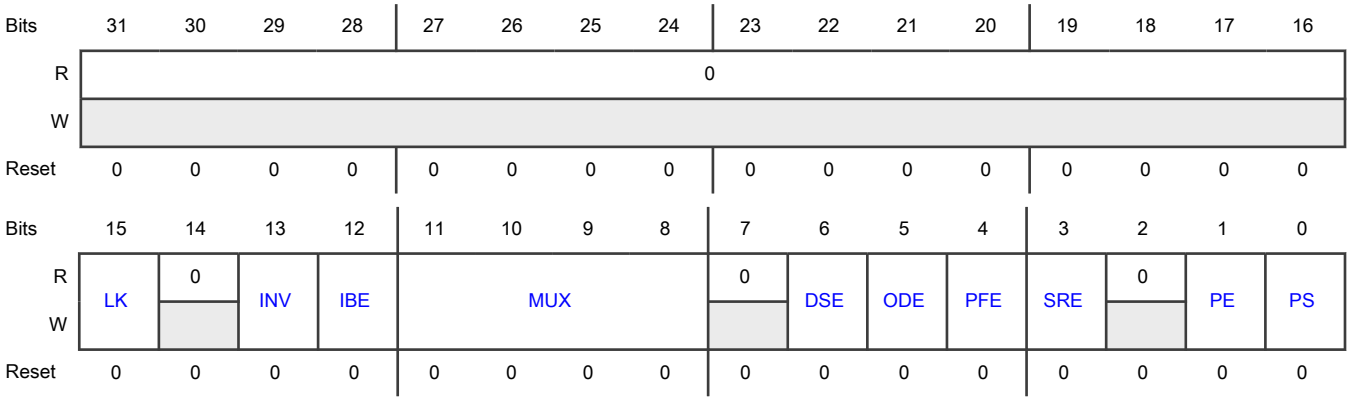
Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	PCR17	—
PORT1	PCR17	—
PORT2	—	PCR17
PORT3	PCR17	—
PORT4	PCR17	—
PORT5	—	PCR17

Diagram



Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The corresponding pin is configured according to the following pin multiplexing slots: 0000b - Alternative 0 (GPIO) 0001b - Alternative 1 (chip-specific) 0010b - Alternative 2 (chip-specific) 0011b - Alternative 3 (chip-specific) 0100b - Alternative 4 (chip-specific) 0101b - Alternative 5 (chip-specific) 0110b - Alternative 6 (chip-specific) 0111b - Alternative 7 (chip-specific) 1000b - Alternative 8 (chip-specific) 1001b - Alternative 9 (chip-specific) 1010b - Alternative 10 (chip-specific) 1011b - Alternative 11 (chip-specific) 1100b - Alternative 12 (chip-specific) 1101b - Alternative 13 (chip-specific)
7 —	Reserved
6 DSE	Drive Strength Enable Configures drive strength, low or high, on each pin. The drive strength configuration is valid for all digital pin multiplexing modes. • When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output. • When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output. 0b - Low 1b - High
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Enables</td></tr>
<tr><td>4
PFE</td><td>Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the passive input filter is disabled on the corresponding pin.
• When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.
See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>PCR17</td><td>—</td></tr><tr><td>PORT1</td><td>PCR17</td><td>—</td></tr><tr><td>PORT3</td><td>—</td><td>PCR17</td></tr><tr><td>PORT4</td><td>—</td><td>PCR17</td></tr></table>0b - Disables
1b - Enables</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
0b - Fast
1b - Slow</td></tr>
<tr><td>2
—</td><td>Reserved</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. 0b - Disables 1b - Enables
0 PS	Pull Select Enables the internal pullup or pulldown resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. • When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. 0b - Enables internal pulldown resistor 1b - Enables internal pullup resistor

75.6.1.21 Pin Control a (PCR18 - PCR19)

Offset

Register	Offset
PCR18	C8h
PCR19	CCh

Function

Configures pin control features on each pin.

NOTE

Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

NOTE

Each module instance supports a different number of registers.

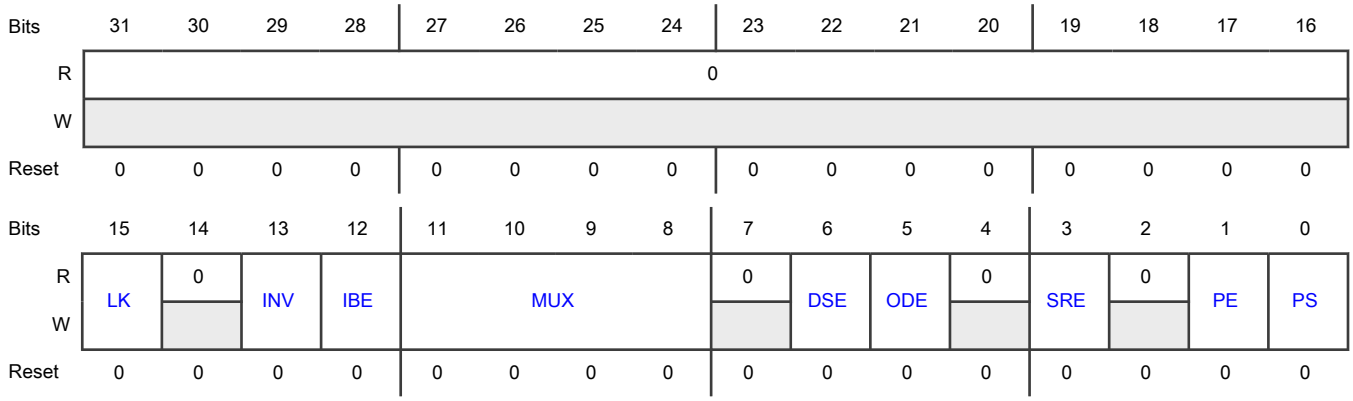
Instance	Register supported	Register not supported
PORT0	PCR18–PCR19	—
PORT1	PCR18–PCR19	—
PORT2	—	PCR18–PCR19

Table continues on the next page...

Table continued from the previous page...

Instance	Register supported	Register not supported
PORT3	PCR18–PCR19	—
PORT4	PCR18–PCR19	—
PORT5	—	PCR18–PCR19

Diagram



Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12	Input Buffer Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
IBE	Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots: 0000b - Alternative 0 (GPIO) 0001b - Alternative 1 (chip-specific) 0010b - Alternative 2 (chip-specific) 0011b - Alternative 3 (chip-specific) 0100b - Alternative 4 (chip-specific) 0101b - Alternative 5 (chip-specific) 0110b - Alternative 6 (chip-specific) 0111b - Alternative 7 (chip-specific) 1000b - Alternative 8 (chip-specific) 1001b - Alternative 9 (chip-specific) 1010b - Alternative 10 (chip-specific) 1011b - Alternative 11 (chip-specific) 1100b - Alternative 12 (chip-specific) 1101b - Alternative 13 (chip-specific)
7 —	Reserved
6 DSE	Drive Strength Enable Configures drive strength, low or high, on each pin. The drive strength configuration is valid for all digital pin multiplexing modes. • When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output. • When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output. 0b - Low 1b - High

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables 1b - Enables
4 —	Reserved
3 SRE	Slew Rate Enable Configures the slew rate feature, fast or slow, on each corresponding pin. The slew rate configuration is valid for all digital pin multiplexing modes. 0b - Fast 1b - Slow
2 —	Reserved
1 PE	Pull Enable Enables the internal pull resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pull resistor is not enabled on the corresponding pin. • When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. 0b - Disables 1b - Enables
0 PS	Pull Select Enables the internal pullup or pulldown resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. • When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. 0b - Enables internal pulldown resistor

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enables internal pullup resistor

75.6.1.22 Pin Control 20 (PCR20)

Offset

Register	Offset
PCR20	D0h

Function

Configures pin control features on each pin.

NOTE

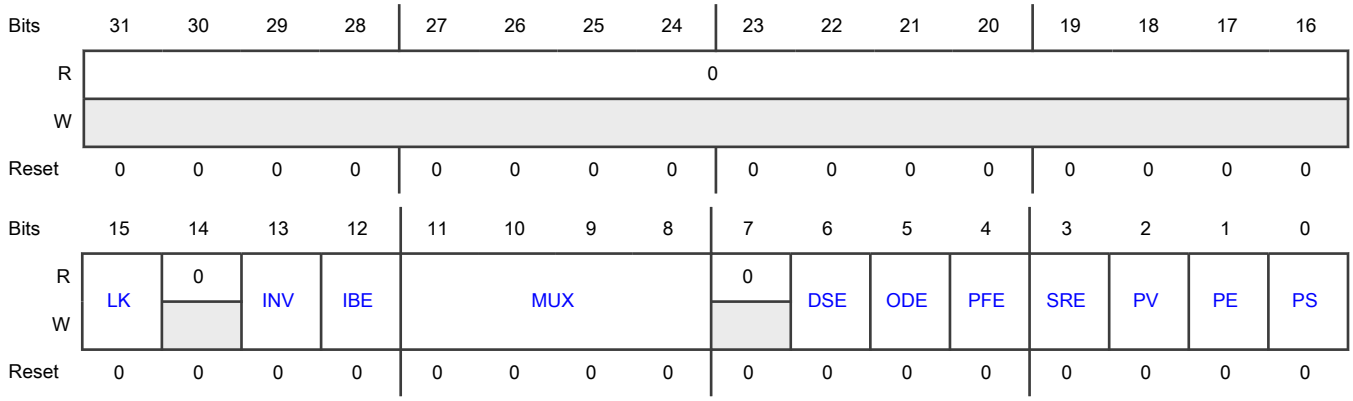
Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	PCR20	—
PORT1	PCR20	—
PORT2	—	PCR20
PORT3	PCR20	—
PORT4	PCR20	—
PORT5	—	PCR20

Diagram



Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The corresponding pin is configured according to the following pin multiplexing slots: 0000b - Alternative 0 (GPIO) 0001b - Alternative 1 (chip-specific) 0010b - Alternative 2 (chip-specific) 0011b - Alternative 3 (chip-specific) 0100b - Alternative 4 (chip-specific) 0101b - Alternative 5 (chip-specific) 0110b - Alternative 6 (chip-specific) 0111b - Alternative 7 (chip-specific) 1000b - Alternative 8 (chip-specific) 1001b - Alternative 9 (chip-specific) 1010b - Alternative 10 (chip-specific) 1011b - Alternative 11 (chip-specific) 1100b - Alternative 12 (chip-specific) 1101b - Alternative 13 (chip-specific)
7 —	Reserved
6 DSE	Drive Strength Enable Configures drive strength, low or high, on each pin. The drive strength configuration is valid for all digital pin multiplexing modes. • When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output. • When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output. 0b - Low 1b - High
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Enables</td></tr>
<tr><td>4
PFE</td><td>Passive Filter Enable
Enables passive input filter on each pin.
The passive filter configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the passive input filter is disabled on the corresponding pin.
• When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input.
See the chip's data sheet for filter characteristics.
NOTE
This field is not supported in every instance. The following table includes only supported registers.
<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>PCR20</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>PCR20</td></tr><tr><td>PORT3</td><td>—</td><td>PCR20</td></tr><tr><td>PORT4</td><td>—</td><td>PCR20</td></tr></table>0b - Disables
1b - Enables</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
0b - Fast
1b - Slow</td></tr>
<tr><td>2
PV</td><td>Pull Value
Selects high or low internal pull resistor value.
The pull value configuration is valid for all digital pin multiplexing modes.
NOTE
This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Function
	<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PORT0</td><td>PCR20</td><td>—</td></tr><tr><td>PORT1</td><td>—</td><td>PCR20</td></tr><tr><td>PORT3</td><td>—</td><td>PCR20</td></tr><tr><td>PORT4</td><td>—</td><td>PCR20</td></tr></table> 0b - Low 1b - High
1 PE	Pull Enable Enables the internal pull resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pull resistor is not enabled on the corresponding pin. • When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. 0b - Disables 1b - Enables
0 PS	Pull Select Enables the internal pullup or pulldown resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. • When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. 0b - Enables internal pulldown resistor 1b - Enables internal pullup resistor

75.6.1.23 Pin Control 21 (PCR21)

Offset

Register	Offset
PCR21	D4h

Function

Configures pin control features on each pin.

NOTE

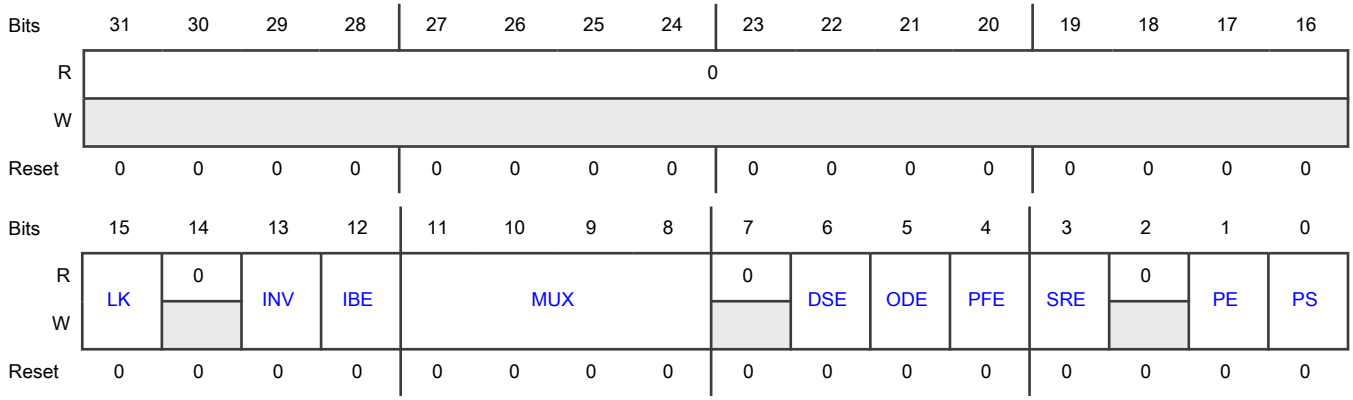
Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	PCR21	—
PORT1	PCR21	—
PORT2	—	PCR21
PORT3	PCR21	—
PORT4	PCR21	—
PORT5	—	PCR21

Diagram



Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die. The corresponding pin is configured according to the following pin multiplexing slots: 0000b - Alternative 0 (GPIO) 0001b - Alternative 1 (chip-specific) 0010b - Alternative 2 (chip-specific) 0011b - Alternative 3 (chip-specific) 0100b - Alternative 4 (chip-specific) 0101b - Alternative 5 (chip-specific) 0110b - Alternative 6 (chip-specific) 0111b - Alternative 7 (chip-specific) 1000b - Alternative 8 (chip-specific) 1001b - Alternative 9 (chip-specific) 1010b - Alternative 10 (chip-specific) 1011b - Alternative 11 (chip-specific) 1100b - Alternative 12 (chip-specific) 1101b - Alternative 13 (chip-specific)

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 —	Reserved
6 DSE	Drive Strength Enable Configures drive strength, low or high, on each pin. The drive strength configuration is valid for all digital pin multiplexing modes. • When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output. • When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output. 0b - Low 1b - High
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables 1b - Enables
4 PFE	Passive Filter Enable Enables passive input filter on each pin. The passive filter configuration is valid for all digital pin multiplexing modes. • When this field = 0, the passive input filter is disabled on the corresponding pin. • When this field = 1, the passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input. See the chip's data sheet for filter characteristics. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PORT0	PCR21	—
PORT1	—	PCR21

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>PORT3</td><td>—</td><td>PCR21</td></tr>
<tr><td>PORT4</td><td>—</td><td>PCR21</td></tr>
</table>
0b - Disables
1b - Enables</td></tr>
<tr><td>3
SRE</td><td>Slew Rate Enable
Configures the slew rate feature, fast or slow, on each corresponding pin.
The slew rate configuration is valid for all digital pin multiplexing modes.
0b - Fast
1b - Slow</td></tr>
<tr><td>2
—</td><td>Reserved</td></tr>
<tr><td>1
PE</td><td>Pull Enable
Enables the internal pull resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pull resistor is not enabled on the corresponding pin.
• When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0b - Disables
1b - Enables</td></tr>
<tr><td>0
PS</td><td>Pull Select
Enables the internal pullup or pulldown resistor.
This configuration is valid for all digital pin multiplexing modes.
• When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
• When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCRn.PE field = 1.
0b - Enables internal pulldown resistor
1b - Enables internal pullup resistor</td></tr>
</table>

75.6.1.24 Pin Control a (PCR22 - PCR31)

Offset

For a = 22 to 31:

Register	Offset
PCRa	80h + (a × 4h)

Function

Configures pin control features on each pin.

NOTE

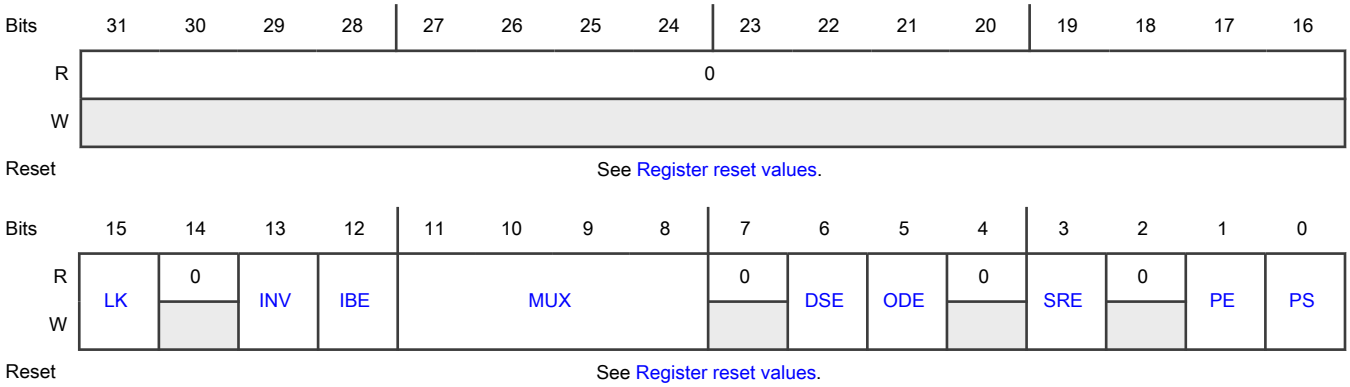
Do not modify pin configuration registers associated with pins that are unavailable in your selected package. All unbonded pins unavailable in your package default to the Disabled state for lowest power consumption.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
PORT0	PCR22–PCR31	—
PORT1	PCR22–PCR23 PCR30–PCR31	PCR24–PCR29
PORT2	—	PCR22–PCR31
PORT3	PCR22–PCR23	PCR24–PCR31
PORT4	PCR22–PCR23	PCR24–PCR31
PORT5	—	PCR22–PCR31

Diagram



Register reset values

Register	Reset value
PCR22–PCR23	PORT0,PORT1: 0000_0000h PORT2: Register not supported PORT3,PORT4: 0000_0000h PORT5: Register not supported
PCR24–PCR29	0000_0000h
PCR30–PCR31	PORT0,PORT1: 0000_0000h PORT2–PORT5: Register not supported

Fields

Field	Function
31-16 —	Reserved
15 LK	Lock Register Locks this PCR. When a PCR*n* is locked, its fields cannot be updated until the next reset. 0b - Does not lock 1b - Locks
14 —	Reserved
13 INV	Invert Input Inverts the digital input. 0b - Does not invert 1b - Inverts
12 IBE	Input Buffer Enable Enables digital input buffer. When disabled, the digital input is required for analog functions. 0b - Disables 1b - Enables
11-8 MUX	Pin Multiplex Control Configures the multiplexing slots on each pin. Not all pins support all pin multiplexing slots. Unimplemented pin multiplexing slots are reserved. Unimplemented pin multiplexing slots can result in different behaviors, if the unimplemented slot is phantomed (because the module is phantomed on that die) versus unimplemented on the die.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The corresponding pin is configured according to the following pin multiplexing slots: 0000b - Alternative 0 (GPIO) 0001b - Alternative 1 (chip-specific) 0010b - Alternative 2 (chip-specific) 0011b - Alternative 3 (chip-specific) 0100b - Alternative 4 (chip-specific) 0101b - Alternative 5 (chip-specific) 0110b - Alternative 6 (chip-specific) 0111b - Alternative 7 (chip-specific) 1000b - Alternative 8 (chip-specific) 1001b - Alternative 9 (chip-specific) 1010b - Alternative 10 (chip-specific) 1011b - Alternative 11 (chip-specific) 1100b - Alternative 12 (chip-specific) 1101b - Alternative 13 (chip-specific)
7 —	Reserved
6 DSE	Drive Strength Enable Configures drive strength, low or high, on each pin. The drive strength configuration is valid for all digital pin multiplexing modes. • When this field = 0, low drive strength is configured on the corresponding pin, if the pin is configured as a digital output. • When this field = 1, high drive strength is configured on the corresponding pin, if the pin is configured as a digital output. 0b - Low 1b - High
5 ODE	Open Drain Enable Enables open drain output on each pin. The open drain configuration is valid for all digital pin multiplexing modes. • When this field = 0, the open drain output is disabled on the corresponding pin. • When this field = 1, the open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. 0b - Disables

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enables
4 —	Reserved
3 SRE	Slew Rate Enable Configures the slew rate feature, fast or slow, on each corresponding pin. The slew rate configuration is valid for all digital pin multiplexing modes. 0b - Fast 1b - Slow
2 —	Reserved
1 PE	Pull Enable Enables the internal pull resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pull resistor is not enabled on the corresponding pin. • When this field = 1, the internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. 0b - Disables 1b - Enables
0 PS	Pull Select Enables the internal pullup or pulldown resistor. This configuration is valid for all digital pin multiplexing modes. • When this field = 0, the internal pulldown resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. • When this field = 1, the internal pullup resistor is enabled on the corresponding pin, if the corresponding PCR*n*.PE field = 1. 0b - Enables internal pulldown resistor 1b - Enables internal pullup resistor

Chapter 76
PDM Microphone Interface (MICFIL)

76.1 Chip-specific MICFIL information

Table 714. Reference links to related information

Topic	Related module	Reference
Full description	MICFIL	MICFIL
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

76.1.1 Module instances

This device has one instance of the MICFIL module, MICFIL0.

76.2 Overview

MICFIL delivers audio from microphones to the processor in several applications, such as mobile telephones. As current digital audio systems use a multi-bit audio signal (also known as multi-bit PCM) to represent the signal, this module implements the required digital interface (a series of filters) to transform a pulse density modulated (PDM) microphone bitstream into a 24-bit PCM signal in the audio band, at a configurable output sampling rate.

The implementation of this digital interface is based on the application of digital signal processing techniques used in hardware. The MICFIL architecture is designed to save gate count and minimize power consumption.

You can implement MICFIL to work in a multi-channel mode. All channels have the same configuration, but you could independently turn on or turn off each input channel.

76.2.1 Block diagram

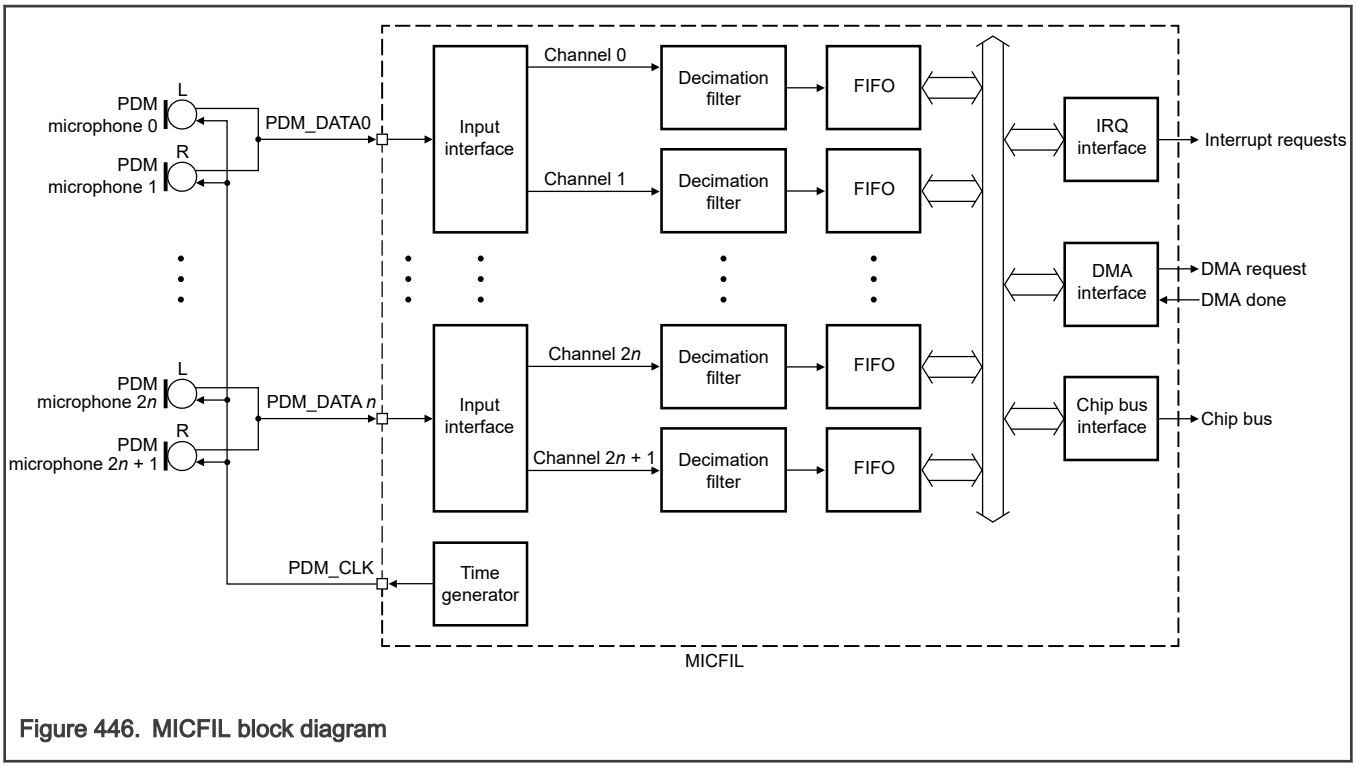


Figure 446. MICFIL block diagram

76.2.2 Features

- Decimation filters:
  - Fixed-point filtering
  - 24-bit PCM audio output
  - Internal clock divider for a programmable PDM clock generation: clock divider bypass capability for low-frequency operations
  - Frame synchronization
  - Full or partial set of channel operations with individual enable controls
  - Programmable decimation rate
  - Programmable DC remover at output
  - Range adjustment capability
- FIFOs with interrupt and DMA capability: each FIFO having a length of 16 entries

76.3 Functional description

76.3.1 Time generator

The time generator unit:

- Generates the PDM_CLK to microphones. This clock is the same and is active for all the PDM microphones, which means, it is not possible to turn off the PDM_CLK only for one single microphone.
- Delivers the PDM_CLK to all microphones that must operate at the same clock frequency. Each input interface receives a time multiplexed PDM bitstream from two PDM microphones and it separates audio information in two channels: left (0) and right (1). Every decimation filter, corresponding to its channel, does this processing.

There exists an internal clock divider (see CTRL_2[CLKDIV]) within the time generator that is enabled when CTRL_2[CLKDIVDIS] = 0; otherwise, the application clock directly drives the microphones.

NOTE

A soft reset affects the time generator unit (see CTRL_1[SRES]). The PDM_CLK starts with a rising edge after a soft reset is issued.

76.3.1.1 Clock divider

The sampling rate of the microphone input depends on the PDM_CLK rate, which you can trim using an internal clock divider, configuring the quality mode (see Quality modes) or trimming the MICFIL's root clock frequency (MICFIL_CLK_ROOT rate).

In case the clock divider is enabled, CTRL_2[CLKDIV] defines the internal clock divider frequency. You must configure CTRL_2[CLKDIV] by using Equation 49.

$$CLKDIV = \frac{MICFIL_CLK_ROOT\ rate}{8 \times OSR \times (output\ rate)}$$

Equation 49. CLKDIV value calculation

After calculating the value of CTRL_2[CLKDIV], you can estimate the generated PDM_CLK rate based on Equation 50, where the "K" factor value depends on the quality mode shown in Table 715 and floor(x) is the function that takes a real number x as input and gives the greatest integer less than or equal to x as output.

$$PDM_CLK\ rate = \frac{MICFIL_CLK_ROOT}{2 \times floor(K \times CLKDIV)}$$

Equation 50. PDM_CLK rate calculation

Table 715. K factor value

Quality mode	K factor
High Quality	1/2
Medium Quality, Very-Low Quality 0	1
Low Quality, Very-Low Quality 1	2
Very-Low Quality 2	4

76.3.1.2 I/O timing requirements

The PDM microphones must meet the setup and hold timing requirements shown in Table 716 and Figure 447. The "K" factor value in Table 716 depends on the quality mode you select based on Table 715.

NOTE

These timing requirements apply only if the clock divider functionality is enabled (see CTRL_2[CLKDIV]) with (CTRL_2[CLKDIVDIS] = 0); otherwise, there are no special timing requirements.

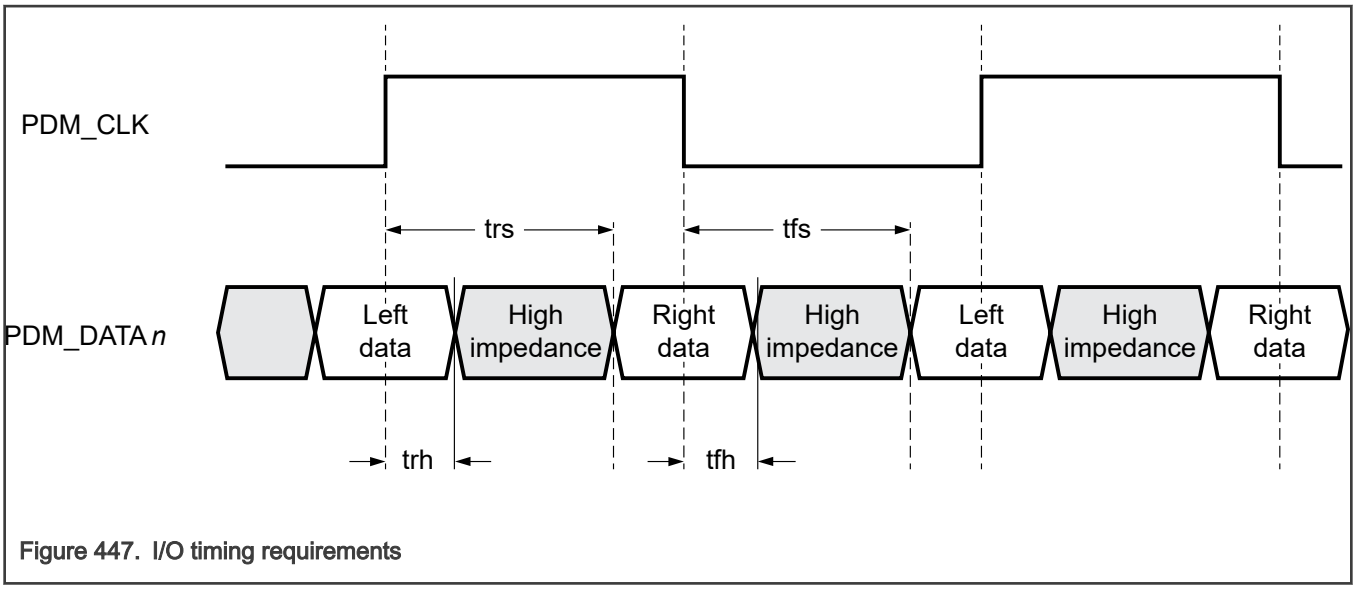


Figure 447. I/O timing requirements

Table 716. Timing parameters

Parameter	Value
trs, tfs	$\leq \frac{floor(K \times CLKDIV) - 1}{MICFIL_CLK_ROOT\ rate}$ Depending on the value of "K," you must ensure that "floor(K x CTRL_2[CLKDIV])" > 1 to avoid timing problems.
trh, tfh	≥ 0

76.3.2 Input interface

Figure 448 shows the timing diagram of the input interface signals when the clock divider is enabled. The bitstream incoming from the microphone data input "n" (PDM_DATA*n*) in the first half (right microphone) of the PDM_CLK is directed to channel "2n+1" and the data generated during the last half (left microphone) is directed to channel "2n".

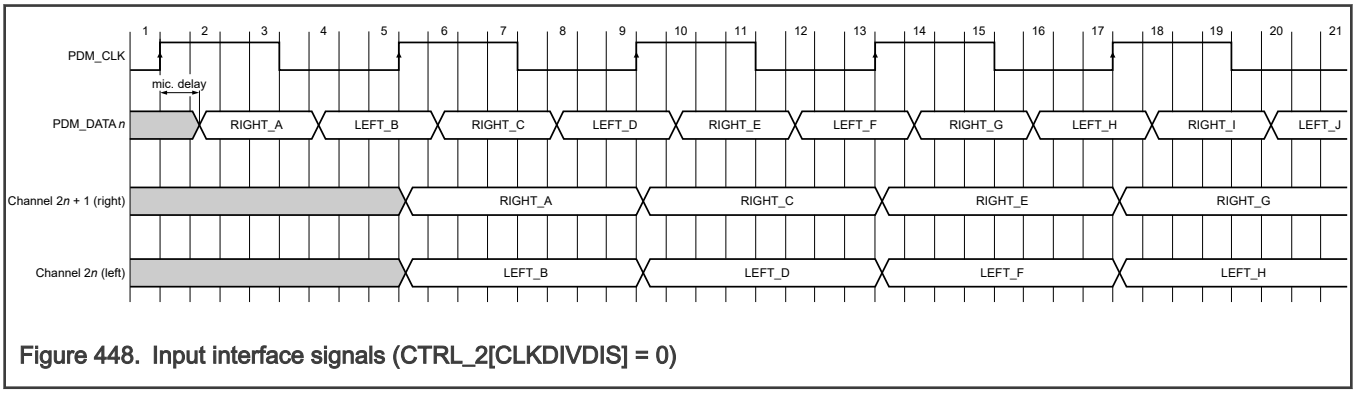


Figure 448. Input interface signals (CTRL_2[CLKDIVDIS] = 0)

76.3.3 Decimation filter

A decimation filter facilitates PDM to PCM conversion.

Figure 449 shows the block diagram of the decimation filter. The block consists of:

- A cascade integrator comb (CIC) filter, which is a low-pass filter that you commonly use to filter sigma-delta modulator output signals. The filter is implemented in the audio band (20 Hz–20.0 kHz @48 kHz output sampling rate by default) with a configurable decimation rate. You can implement it using a series of CIC, compensation, and DC remover filters.

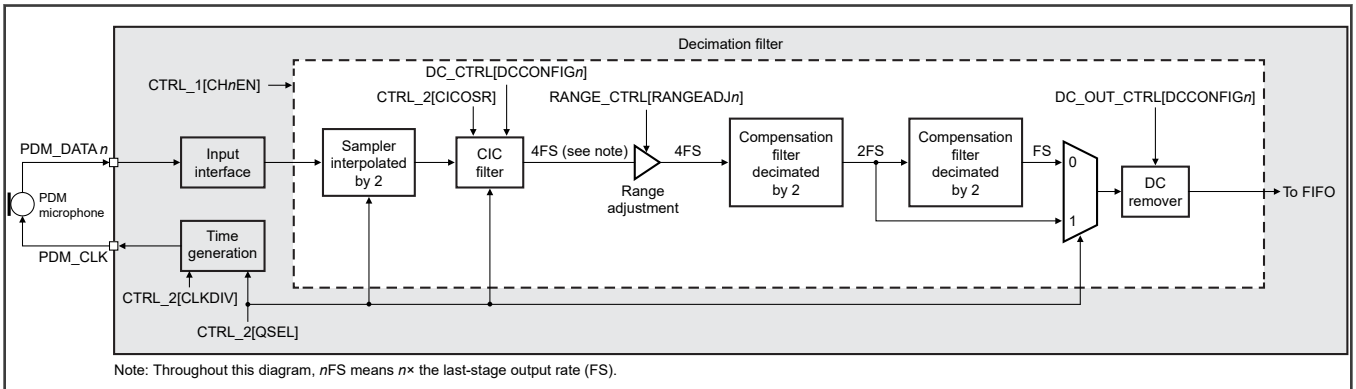
- A DC remover, which is a high-pass filter that you use to remove the DC component of the processed signal with a configurable cut-off frequency. Two DC removers exist: one in the input of the decimation filter, embedded in the CIC filter, and another in the last stage of the decimation filter.
- Two compensation filters for each channel, which implement a low-pass digital filter with decimation by 2. You can use them to compensate the high CIC drop in passband. These filters help the FIR filter block to flatten the overall passband response.

You activate the "Sampler" block depending on the quality mode selected (using CTRL_2[QSEL]). In Low-Quality and Very-Low-Quality 2 modes, the sampling rate is doubled.

The decimation filter output depends on the quality mode that you select by using CTRL_2[QSEL], based on different compensation filter output. You can supply the DC remover with the 2 FS signal, instead of the 1 FS signal, to achieve lower-power consumption, bypassing the second compensation decimation filter. The second compensation filter is bypassed when CTRL_2[QSEL] = 110b, 101b, or 100b (in Very-Low Quality modes).

You can adjust the range of each decimation filter by using a range adjustment block so that you can raise an error interrupt in case an adjustment overflow or underflow is detected.

The output of a decimation filter is stored into a FIFO buffer, and each FIFO is mapped to MICFIL Output Result (DATACH0 - DATACH3). It is possible to generate either an interrupt or a DMA request when, in each FIFO of all enabled channels, the number of data stored surpasses a configured watermark.



Note: Throughout this diagram, *n*FS means *n*× the last-stage output rate (FS).

Figure 449. Decimation filter block diagram

76.3.3.1 CIC filter

The CIC filter is an optimized implementation of the low-pass SINC filter, with multiple stages. It is widely used because it requires just adders and subtracters. The CIC filter comprises an integrator and a comb as its sub-blocks. As shown in the following figure, CTRL_2[CICOSR] and the quality mode you select (see Quality modes) define the CIC decimation rate.

$$OSR = 16 - CICOSR$$

$$CIC\ decimation\ rate = \begin{cases} 2 \times OSR & ; If\ HQ,\ VLQ0 \\ OSR & ; others \end{cases}$$

Equation 51. CIC decimation rate

76.3.3.2 DC remover

Two DC removers exist in the filter chain, one in the input and another in the output. The former is integrated in the CIC filter, and the latter is placed just before the FIFO. Both can be bypassed independently.

The DC remover eliminates the DC level component of the signal using a high-pass filter with a very low cut-off frequency that you select using DC_CTRL[DCCONFIG*n*]. This filter is bypassed when the value of this field is 11b.

NOTE

See PARAM[DC_OUT_BYPASS] and PARAM[DC_BYPASS] to know whether one of the DC removers is permanently disabled in the chip.

76.3.3.3 Compensation filters

A compensation filter divides the baseband signal into two equal subbands before it could to be decimated by 2. It also compensates the CIC filter drop in passband and helps to flatten the overall passband response. MICFIL includes two compensation filter stages, with the possibility of bypassing the last one.

76.3.3.4 Filter performance

76.3.3.4.1 Overall filter gain

As shown in Equation 52, the overall filter gain depends on:

- The quality mode you select (see Quality modes).
- The CIC decimation rate (see Equation 51).
- Dynamic range adjustment of the CIC filter (see RANGE_CTRL[RANGEADJ*n*]).

NOTE

To get a better audio performance, you must configure and adjust RANGE_CTRL[RANGEADJ*n*] depending on the quality mode selected, as pointed out in the register description.

$$Overall\ filter\ gain\ (dB) = \begin{cases} 100 \times log_{10}(32 - 2 \times CICOSR) + 6.02 \times RANGE_CTRL[RANGEADJn] - 150.50 & ; if\ QSEL \in [HQ,VLQ0] \\ 100 \times log_{10}(16 - CICOSR) + 6.02 \times RANGE_CTRL[RANGEADJn] - 150.50 & ; others \end{cases}$$

Equation 52. Overall filter gain

Finally, the input and output dBFS levels relate to each other, as shown in Equation 53.

$$Output\ (dBFS) = input\ (dBFS) + overall\ filter\ gain\ (dB)$$

Equation 53. Filter input and output relationship

76.3.3.4.2 Overall passband

The decimation filter passband depends on the quality mode and output rate (FS). For instance, the overall filter passband for different output data rates when MICFIL is in High (HQ), Medium (MQ), and Low-Quality (LQ) modes is shown in Table 717.

Table 717. Filter passband in HQ, MQ, and LQ modes (DC_CTRL[DCCONFIG*n*] = 0)

Output data rate	Overall filter passband
FS	**0.0004FS - 0.4167FS**
16 kHz	6.67 Hz - 6.67 kHz
48 kHz	20 Hz - 20 kHz

The overall filter passband for different output data rates when MICFIL is in Very-Low Quality (VLQ) modes is shown in Table 718.

Table 718. Filter passband in VLQn modes (DC_CTRL[DCCONFIG*n*] = 0)

Output data rate	Overall filter passband
FS	**0.0002FS - 0.2083FS**
16 kHz	3.33 Hz - 3.33 kHz
48 kHz	10.00 Hz - 10.00 kHz
96 kHz	20 Hz - 20 kHz

76.3.3.5 Frame synchronization

If CTRL_1[FSYNCEN] = 1, MICFIL sends data to FIFOs in frame size defined by FSYNC[FSYNCLEN]. In this way, if a channel is enabled while MICFIL is running (CTRL_1[PDMIEN] = 1), MICFIL starts sending data to its respective FIFO after the current frame is sent completely. If the channel is disabled:

- if at least one other channel is still enabled, the corresponding channel stops sending data after the current frame is sent completely.
- if after that no channel is enabled, MICFIL stops sending data immediately, even if the act of sending frames is incomplete.

Below example shows the above behavior when two channels, DATACH0 and DATACH2 are enabled, and then disabled one after the other.

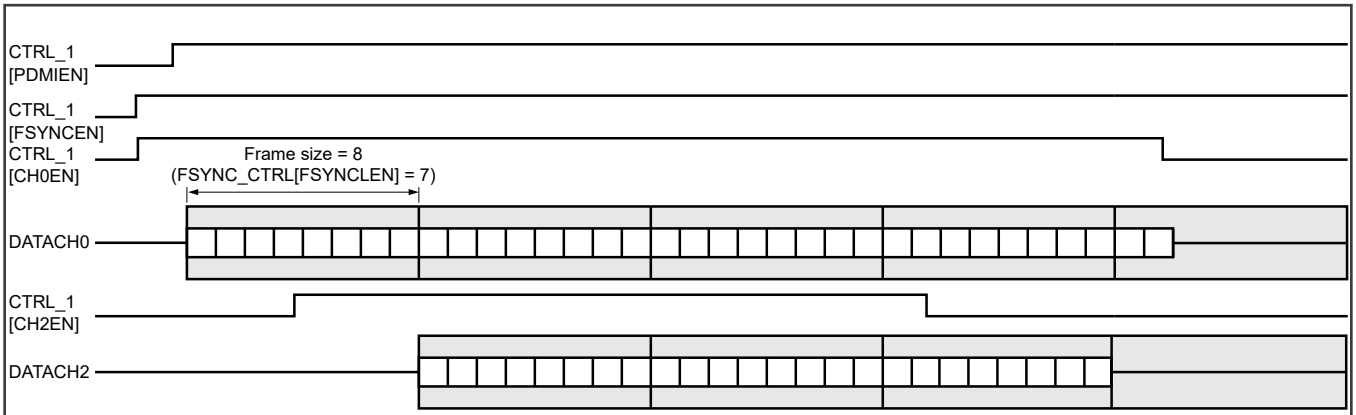


Figure 450. Frame synchronization example

76.3.3.6 Quality modes

If the decimation filter is running independently on the selected power mode, MICFIL can operate in multiple different quality modes that you can select by using CTRL_2[QSEL]. The decimation and interpolation rates in internal blocks are shown in Table 719, where the oversampling rate (OSR) depends on the value of CTRL_2[CICOSR], as shown in the following equation.

$$OSR = 16 - CICOSR$$

Equation 54. OSR

You can configure the selected quality dynamically when the decimation filter is running; the output rate is maintained in every quality selected.

The passband of the decimation filter depends on the quality mode (see Overall passband).

To maintain the same output rate in the decimation filter when the quality is changed, the PDM_CLK rate changes depending on the quality selected. For more on PDM_CLK, see Time generator.

You can enable or disable the sampler and compensation filters according to the selected quality. When they are disabled, you can achieve a low-power consumption. The selected quality also determines whether the CIC filter decimation rate is doubled. For more, see Decimation filter.

With an increase of the clock rate and OSR, you can expect a better signal-to-noise ratio (SNR) performance, which in turn leads to a higher audio quality.

Table 719. Quality modes

Quality mode	CTRL_2[QSEL]	Sampler interpolation	CIC filter decimation	First-compensation filter decimation	Second-compensation filter decimation	PDM_CLK rate	Passband
High Quality	001	—	:(2OSR)	:2	:2	Output rate × 8 × OSR	To ~0.5 × output rate
Medium Quality	000	—	:OSR	:2	:2	Output rate × 4 × OSR	To ~0.5 × output rate
Low Quality	111	× 2	:OSR	:2	:2	Output rate × 2 × OSR	To ~0.5 × output rate
Very-Low Quality 0	110	—	:(2OSR)	:2	—	Output rate × 4 × OSR	To ~0.25 × output rate
Very-Low Quality 1	101	—	:OSR	:2	—	Output rate × 2 × OSR	To ~0.25 × output rate
Very-Low Quality 2	100	× 2	:OSR	:2	—	Output rate × OSR	To ~0.25 × output rate

76.3.3.7 Output rate

The output rate is the sampling rate at the decimation filter output. It depends on the sampling rate of the microphone input and the configured quality mode (see Quality modes). Common audio sampling rates are 48 kHz and 16 kHz. In this chapter, the output rate is sometimes referred to as FS.

76.3.4 Bus interface

The bus interface manages DMA transactions, interrupt requests, and user register interface with the chip.

MICFIL can deliver an interrupt request or DMA request. Then, the chip or DMA, respectively, could access the filter results stored in FIFOs via the internal bus interface.

For detailed information on DMA transactions and interrupt requests, see DMA and Interrupts.

76.3.4.1 FIFO

This is the first-in, first-out (FIFO) buffer that saves the overall filtering results. The length of each FIFO is 16 entries, and there exists one FIFO for each channel. A push writes data into the FIFO, and a read pops data from FIFO. The push and pop operations from FIFO are in different clock domains. The push operation uses the MICFIL clock while the pop operation uses the CPU clock. The FIFO has a configurable watermark (see FIFO_CTRL[FIFOWMK]). When the amount of data in all FIFOs of the enabled channels is greater than the watermark value (FIFO_CTRL[FIFOWMK]), a DMA or IRQ is requested, depending on the value of CTRL_1[DISEL]. After CTRL_1[PDMIEN] enables the filters, the initial FIR results are discarded and not written into the FIFOs. Approximately 68 initial FIR results are not stored in FIFOs. After the filter generates valid data, the results are pushed into the FIFO.

A new filtered result is discarded (not stored) when the FIFO is full and CTRL_1[FIFOOVF*n*] becomes 1. Similarly, when trying a pop operation, if the FIFO is empty, no valid data is read and CTRL_1[FIFOUND*n*] becomes 1. In both cases, when overflow

or underflow occurs, they are flagged in MICFIL FIFO Status (FIFO_STAT) to the respective channel, and an error interrupt is generated if CTRL_1[ERREN] is 1.

When you reset the FIFO by writing 1 to CTRL_1[SRES], zero values fill all FIFO positions, as well as push and pop pointers are set to their initial positions.

76.3.5 Modes of operation

This section describes the MICFIL operation modes that depend on:

- Whether MICFIL receives a Stop request.
- The chip's Debug mode.
- The configuration of CTRL_1[MDIS], CTRL_1[DBGE], CTRL_1[DBG],and CTRL_1[DOZEN], as summarized in the following table.

Table 720. Mode selection

Mode	Normal	Debug		Low-Power		Disable/Low-Leakage (DLL)		
CTRL_1[MDIS]	0	0		0		0		1
Doze request and CTRL_1[DOZEN] = 1	0	0		0		0	1	Any
Stop request	0	0		1		1		Any
CTRL_1[DOZEN]	Any	Any		0		1		Any
CTRL_1[DECFILS]	Any	Any		1	0	Any		
Debug request or CTRL_1[DBG]	0	1		Any		Any		
CTRL_1[DBGE]	Any	1	0	Any		Any		
CPU clock	Enabled	Enabled		Gated (Externally)		Gated (Externally)	Enabled	Gated (Internally)
MICFIL internal clock	Enabled	Enabled		Enabled		Gated		
Decimation filter state	Run[1]	Run[1]	Stop	Run[1]	Stop	Stop		

1. Depends on the configuration of CTRL_1[PDMIEN] and CTRL_1[CH*n*EN].

76.3.5.1 Normal mode

This is the default operational mode for MICFIL. In this mode, you can enable the channel functionality to perform the filtering operation.

76.3.5.2 Debug mode

You could use this mode to check the status of the module, when requested. In this mode, you can stop MICFIL after the remaining data processing is complete, depending on the value of CTRL_1[DBGE]. You can activate this mode by writing 1 to CTRL_1[DBG].

If CTRL_1[DBGE] = 0, and CTRL_1[DBG] = 1 when MICFIL is running, MICFIL stops after processing the remaining data. Otherwise, if [DBGE] = 1, MICFIL continues to run.

MICFIL cannot enter Debug mode after entering DLL or Low-Power mode.

The Debug mode affects all channels in MICFIL. Access to output buffers, as well as their related flags, remains operational.

76.3.5.3 Low-Power mode

MICFIL enters Low-Power mode when MICFIL receives a Stop request and CTRL_1[DOZEN] = 0. In Low-Power mode, MICFIL is able to filter the PDM microphone data and can generate a wake-up event through asynchronous DM or IRQ requests, depending on the values of CTRL_1[CH*n*EN] and CTRL_1[PDMIEN].

After MICFIL enters Low-Power mode:

- If CTRL_1[DECFILS] = 1, Stop request is acknowledged immediately and the decimation filter keeps running.
- If CTRL_1[DECFILS] = 0, the decimation filter processes the last input sample. After that, Stop request is acknowledged and the filter is stopped from running.

You can generate DMA and IRQ requests asynchronously to wake up the CPU.

NOTE

The asynchronous DMA or interrupt requests are generated only in this mode.

76.3.5.4 Disable/Low-Leakage (DLL) mode

In this mode, MICFIL stops after the remaining data processing is complete. MICFIL is in DLL mode when:

- MICFIL receives a Stop request and CTRL_1[DOZEN] = 1.
- CTRL_1[MDIS] = 1.

If you request DLL mode when MICFIL is running, MICFIL stops after completion to process the remaining data.

You can write only to CTRL_1 and CTRL_2, with the exception of writes to CTRL_1[DBG] and CTRL_1[SRES], which are ignored. However, you can still read these fields.

76.3.6 Clocking

Table 721. MICFIL clocks

Clock name	Description
CPU clock (MICFIL_CLK, MICFIL_CLK_S)	Clock that controls the registers
MICFIL internal clock (MICFIL_CLK_APP)	Application clock that operates the filter

Besides the above clock signals, MICFIL generates PDM_CLK for PDM microphones based on MICFIL internal clock frequency and the value in CTRL_2[CLKDIV].

76.3.7 Interrupts

If CTRL_1[DISEL] = 10b, an interrupt request indicates that filtering results are stored in the FIFOs to be read by the CPU. In this case, you can read the data in DATACH*n*[DATA], and STAT[CH*n*F] becomes 1 for the enabled channels when the CHn FIFO surpasses the watermark level.

When MICFIL is in Low-Power mode, the CPU clock is disabled, but the MICFIL clock remains enabled. In this case, an asynchronous interrupt is delivered to the system. This wakes up the CPU clock, which returns to Normal mode. The system could read the FIFO data, after which it must clear STAT[CH*n*F] to get a new interrupt request.

An error interrupt request can also be generated when an exceptional condition happens, such as an overflow or underflow in a channel output, or an overflow or underflow of a FIFO buffer.

CTRL_1[ERREN] enables error interruption and the respective status flag signalizes the type of error:

- OUTOVFn and OUTUNFn for channel outputs errors
- FIFO_STAT[FIFOOVF*n*] and FIFO_STAT[FIFOUND*n*] for FIFO errors

76.3.8 DMA

If CTRL_1[DISEL] = 1, the FIFO stored samples are read through DMA transactions. Then if there is at least one enabled channel and all FIFOs of all enabled channels reach their watermark levels, the output interface makes a request for a DMA transaction.

If CTRL_1[PDMIEN] = 1 and the filters start to operate, you must wait until the filter results stabilize and no filter results are stored in the FIFOs. Therefore, the first DMA request takes longer to occur than the ones that follow.

STAT[CH*n*F] becomes 0 after a DMA transaction is complete.

If MICFIL is in Low-Power mode and the watermark level is surpassed, an asynchronous DMA request is delivered to the system. Then, the system wakes up the CPU clock, returning to Normal mode to read the FIFO data.

76.4 External signals

Table 722. PDM microphone interface external signals

Signal	Description	Direction
PDM_DATA*n*	Data bitstream received from the nth PDM microphone pair throughout the input interface (see Input interface)	I
PDM_CLK	Clock that is delivered to the PDM microphones from the clock generator (see Time generator)	O

NOTE

MICFIL does not provide metastability protection or filtering for input data. You must generate the data using the provided PDM_CLK.

76.5 Initialization

76.5.1 Procedure

Perform this procedure for MICFIL initialization (by using only decimation filters):

1. Program CTRL_2[CLKDIV], CTRL_1[CH*n*EN], FIFO_CTRL[FIFOWMK], CTRL_1[DISEL], CTRL_1[ERREN], and so on, as required.
2. Start the module operation by writing 1 to CTRL_1[PDMIEN].

The module is ready to receive data from the input ports of the enabled channels.

76.6 Application information

76.6.1 Reconfiguration procedure

Perform this procedure to reconfigure MICFIL:

1. Write 0 to CTRL_1[PDMIEN].
2. Wait until STAT[BSY_FIL] becomes 0.
3. Run a soft-reset cycle.
4. Run an initialization procedure as described in Procedure.

76.7 MICFIL register descriptions

This section provides the memory map and detailed description of all MICFIL registers. Any access to reserved areas can issue a slave bus error indication.

You must implement the registers allocated in this memory map for up to 2 pairs of microphones, indicating that 4 channels are available.

76.7.1 MICFIL memory map

MICFIL0 base address: 4010_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	MICFIL Control 1 (CTRL_1)	32	RW	0000_0000h
4h	MICFIL Control 2 (CTRL_2)	32	RW	0000_0000h
8h	MICFIL Status (STAT)	32	RW	0000_0000h
10h	MICFIL FIFO Control (FIFO_CTRL)	32	RW	0000_000Fh
14h	MICFIL FIFO Status (FIFO_STAT)	32	RW	0000_0000h
24h - 30h	MICFIL Output Result (DATACH0 - DATACH3)	32	R	0000_0000h
64h	MICFIL DC Remover Control (DC_CTRL)	32	R	0000_00FFh
68h	MICFIL Output DC Remover Control (DC_OUT_CTRL)	32	RW	0000_0000h
74h	MICFIL Range Control (RANGE_CTRL)	32	RW	0000_0000h
7Ch	MICFIL Range Status (RANGE_STAT)	32	RW	0000_0000h
80h	Frame Synchronization Control (FSYNC_CTRL)	32	RW	0000_0000h
84h	Version ID (VERID)	32	R	020F_0000h
88h	Parameter (PARAM)	32	R	0000_0742h

76.7.2 MICFIL Control 1 (CTRL_1)

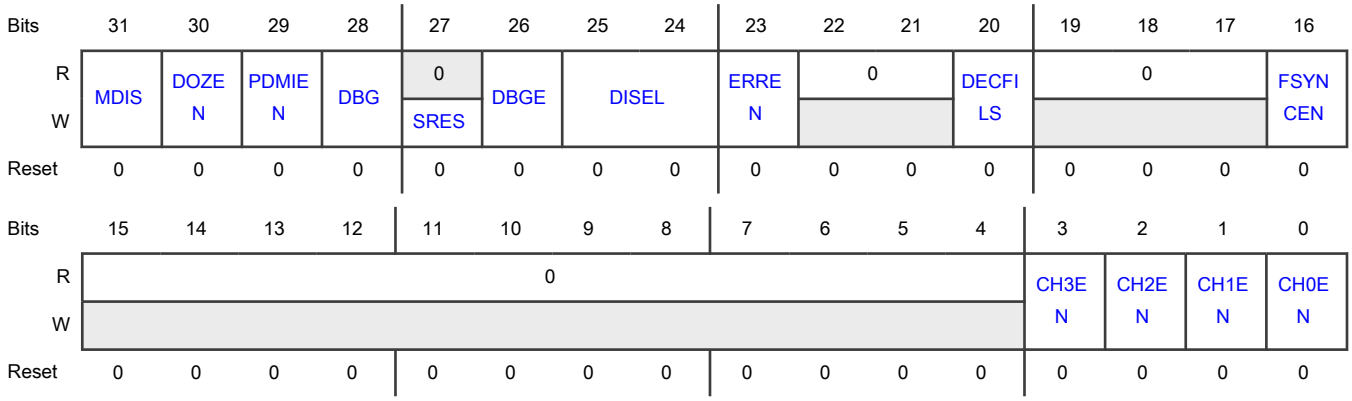
Offset

Register	Offset
CTRL_1	0h

Function

Contains control fields for MICFIL.

Diagram



Fields

Field	Function
31 MDIS	Module Disable Places MICFIL in DLL mode (see Disable/Low-Leakage (DLL) mode). The generated PDM_CLK stops, and consequently, the microphone generates no data. **NOTE** When this field = 1, only writes to MICFIL Control 1 (CTRL_1) and MICFIL Control 2 (CTRL_2) are allowed with the exception of writes to CTRL_1[DBG] and CTRL_1[SRES]. A slave bus error indication is issued if you write to any other register. 0b - Normal mode 1b - DLL mode
30 DOZEN	Stop Enable When MICFIL receives a Stop request and this field is active, MICFIL enters DLL mode (see Disable/Low-Leakage (DLL) mode). 0b - Disables 1b - Enables
29 PDMIEN	MICFIL Enable Starts the module operation. Only the channels that CTRL_1[CH*n*EN] enables start operating. When this field becomes 0, MICFIL stops after it completes the remaining filter processing. **NOTE** You must enable this field as the last step when initializing the module (see Initialization). 0b - Stops MICFIL operation 1b - Starts MICFIL operation
28 DBG	Debug Mode Specifies the mode (Debug or Normal) that MICFIL operates in.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	MICFIL operates in Debug mode, depending on the value of CTRL_1[DBGE]. When Debug mode is disabled, MICFIL operates in Normal mode. See Debug mode for more on Debug mode. **NOTE** You can write to this field only when CTRL_1[MDIS] = 0. 0b - Normal 1b - Debug
27 SRES	Software Reset Provides the CPU with the capability to initialize MICFIL through the slave-bus interface. The field always reads 0, and is effective only when CTRL_1[MDIS] = 0. It is a self-clearing field. **NOTE** You must write 1 to this field only when STAT[BSY_FIL] = 0. You can use this field to clear data in all filters. It resets the following blocks and registers: • All internal buffers in decimation filters • All FIFOs • STAT[CH*n*F] and MICFIL FIFO Status (FIFO_STAT) • Time generator 0b - No action 1b - Software reset
26 DBGE	Module Enable in Debug Enables operation in Debug mode (see Debug mode). 0b - Disables after completing the current frame 1b - Enables operation
25-24 DISEL	DMA Interrupt Selection Specifies the type of interrupt requests (DMA or filter) that MICFIL performs when the number of FIFO elements surpasses the configured watermark. Note that requests are only generated from the channels for which CTRL_1[CH*n*EN] = 1. **NOTE** You must not change the value of this field when STAT[BSY_FIL] = 1. 00b - Disables DMA and interrupt requests 01b - Enables DMA requests 10b - Enables interrupt requests 11b - Reserved
23	Error Interruption Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
ERREN	Enables error interrupts (exceptions). The following conditions can lead to errors: • An overflow or underflow in the FIFOs (indicated by FIFO_STAT[FIFOOVF*n*] or FIFO_STAT[FIFOUND*n*], respectively) • An overflow or underflow in the decimations filters (indicated by RANGE_STAT[RANGEOVF*n*] or RANGE_STAT[RANGEUNF*n*], respectively) **NOTE** You must not change the value of this field when STAT[BSY_FIL] = 1. 0b - Disables 1b - Enables
22-21 —	Reserved
20 DECFILS	Decimation Filter Enable in Stop Enables decimation filters to run when MICFIL receives a Stop request. 0b - Stops decimation filter 1b - Keeps decimation filter running
19-17 —	Reserved
16 FSYNCEN	Frame Synchronization Enable Enables frame synchronization (see Frame synchronization). **NOTE** You must not change the value of this field when STAT[BSY_FIL] = 1. 0b - Disables 1b - Enables
15-4 —	Reserved
3-0 CHnEN	Channel n Enable Enables the decimation filter channel n (see Initialization for more information).

76.7.3 MICFIL Control 2 (CTRL_2)

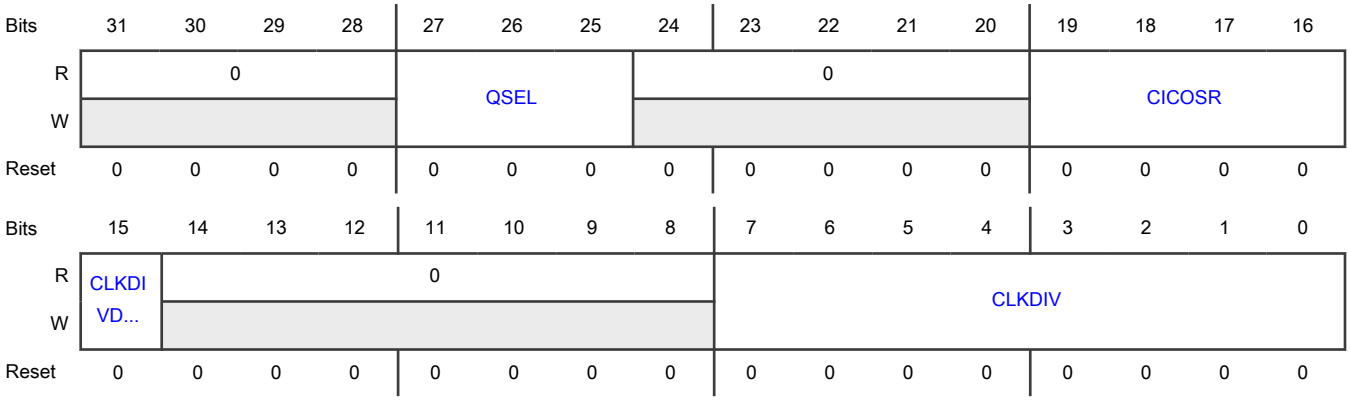
Offset

Register	Offset
CTRL_2	4h

Function

Contains control fields for MICFIL.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-25 QSEL	Quality Mode Defines the actual quality mode (see Quality modes) of the decimation filter. **NOTE** You must not change the value of this field when STAT[BSY_FIL] = 1. 000b - Medium-Quality mode 001b - High-Quality mode 100b - Very-Low-Quality 2 mode 101b - Very-Low-Quality 1 mode 110b - Very-Low-Quality 0 mode 111b - Low-Quality mode
24-20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
19-16 CICOSR	CIC Decimation Rate Configures the CIC filter decimation rate (see Equation 51 for more information). **NOTE** You must not change the value of this field when STAT[BSY_FIL] = 1. 0000b - CIC oversampling rate = 0 0001b - CIC oversampling rate = 1 0010b-1110b - ... 1111b - CIC oversampling rate = 15
15 CLKDIVDIS	Clock Divider Disable Disables and bypasses the internal clock divider (see Clock divider) to drive microphones directly with the application clock. **NOTE** You must not change the value of this field when STAT[BSY_FIL] = 1. 0b - Enables 1b - Disables
14-8 —	Reserved
7-0 CLKDIV	Clock Divider Configures the internal clock divider value (see Clock divider for more information) using the following equation: $$PDM_CLK\ rate = \frac{MICFIL_CLK_ROOT}{2 \times floor(K \times CLKDIV)}$$ **NOTE** • You must not change the value of this field when STAT[BSY_FIL] = 1. • [CLKDIV] = 0 has the same effect as [CLKDIV] = 1. In Medium Quality mode, [CLKDIV] = 0 or 1 has the same effect as [CLKDIV] = 2. In High Quality mode, [CLKDIV] = 0 or 1 is not allowed. 0000_0000b - Internal clock divider value = 0 0000_0001b - Internal clock divider value = 1 0000_0010b-1111_1110b - ... 1111_1111b - Internal clock divider value = 255

76.7.4 MICFIL Status (STAT)

Offset

Register	Offset
STAT	8h

Function

Contains status flags for MICFIL.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BSY_FIL	0	0	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CH3F	CH2F	CH1F	CH0F
W													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 BSY_FIL	Busy Flag Signalizes when at least a decimation filter channel is running and the PDM_CLK is being generated. You could use this field as a confirmation that all decimation filters are stopped in a transition to Debug mode (when CTRL_1[DBGE] = 0) or DLL mode. 0b - MICFIL is stopped 1b - MICFIL is running
30 —	Reserved
29 —	Reserved
28-16 —	Reserved
15-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3-0 CHnF	Channel n Output Data Flag Indicates whether this channel's FIFO has surpassed its watermark value and the data is available in MICFIL Output Result (DATACH0 - DATACH3). This flag contributes to generate an interrupt or DMA request if CTRL_1[DISEL] enables it. Write 1 to CTRL_1[DISEL] to clear this flag. If the DMA request is set (CTRL_1[DISEL] = 1), [CHnF] is set according to the watermark level . In other cases (IRQ included), [CHnF] is set according to the watermark level and is cleared when you write 1 to it. **NOTE** If DMA is enabled, this flag clears itself after the DMA transaction is complete. 0b - Not surpassed 1b - Surpassed

76.7.5 MICFIL FIFO Control (FIFO_CTRL)

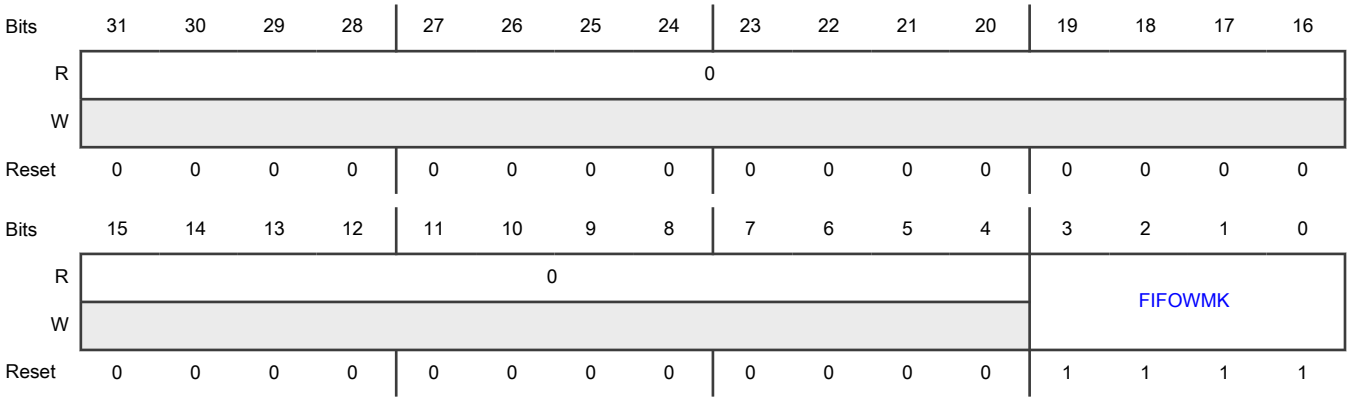
Offset

Register	Offset
FIFO_CTRL	10h

Function

Contains MICFIL FIFO control fields.

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 FIFOWMK	FIFO Watermark Control Controls the watermark of the FIFO used to set STAT[CH*n*F]. If the number of results in the FIFO is greater than the value of this field, STAT[CH*n*F] is set. A DMA request or interrupt can be also generated according to the configuration of CTRL_1[DISEL]. **NOTE** You must not change the value of this field when STAT[BSY_FIL] = 1. Before changing the watermark value, you must service all filtered data and clear channel flags.

76.7.6 MICFIL FIFO Status (FIFO_STAT)

Offset

Register	Offset
FIFO_STAT	14h

Function
Contains MICFIL FIFO status flags.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				FIFOUND3	FIFOUND2	FIFOUND1	FIFOUND0	0				FIFOOVF3	FIFOOVF2	FIFOOVF1	FIFOOVF0
W					W1C	W1C	W1C	W1C					W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 FIFOUNDn	FIFO Underflow Exception Flag for Channel n Indicates an exceptional underflow condition in the FIFO. If the FIFO is empty and a read operation is requested, then this flag is set. 0b - No exception by FIFO underflow 1b - Exception by FIFO underflow
7-4 —	Reserved
3-0 FIFOOVFn	FIFO Overflow Exception Flag for Channel n Indicates an exceptional overflow condition in the FIFO. If the FIFO is full and you receive an additional result to be written, this flag is set. 0b - No exception by FIFO overflow 1b - Exception by FIFO overflow

76.7.7 MICFIL Output Result (DATACH0 - DATACH3)

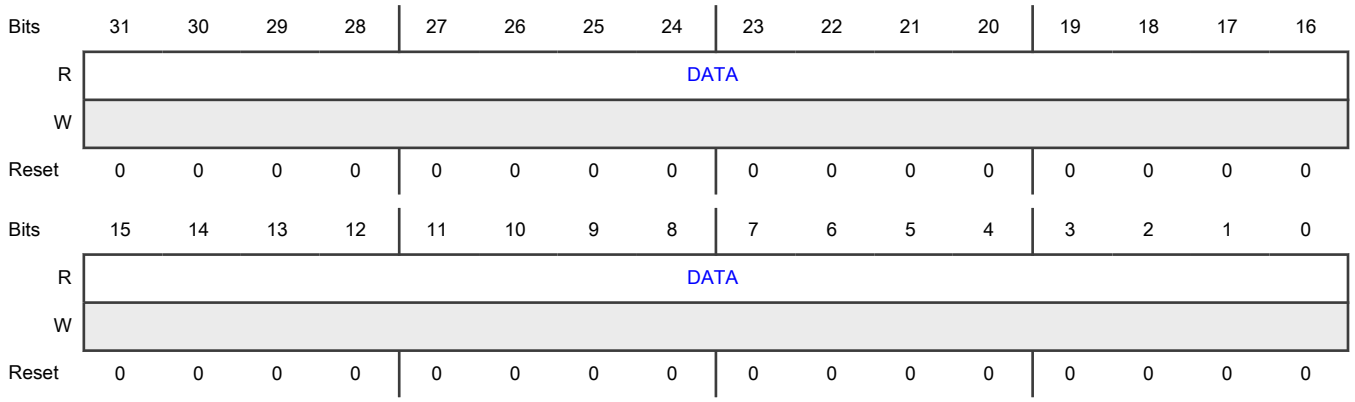
Offset

Register	Offset
DATACH0	24h
DATACH1	28h
DATACH2	2Ch
DATACH3	30h

Function

Contains the word value stored at the top of the channel's n FIFO.

Diagram



Fields

Field	Function
31-0 DATA	Channel n Data Reflects the word value stored at the top of the channel's n FIFO. The samples are integer-signed numbers in the two’s-complement format. **NOTE** Even if the FIFO data width is 32-bit, only the 24 more significant bits have information, and the other bits are always 0.

76.7.8 MICFIL DC Remover Control (DC_CTRL)

Offset

Register	Offset
DC_CTRL	64h

Function
Contains DC remover control fields.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								DCCONFIG3		DCCONFIG2		DCCONFIG1		DCCONFIG0	
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Fields

Field	Function
31-8 —	Reserved
7-6: DCCONFIG3	Channel n DC Remover Configuration Defines the value of the cut-off frequency of the DC remover.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-4: DCCONFIG2 3-2: DCCONFIG1 1-0: DCCONFIG0	00b - 20 Hz (PDM_CLK = 3.072 MHz) 01b - 13.3 Hz (PDM_CLK = 3.072 MHz) 10b - 40 Hz (PDM_CLK = 3.072 MHz) 11b - DC remover is bypassed

76.7.9 MICFIL Output DC Remover Control (DC_OUT_CTRL)

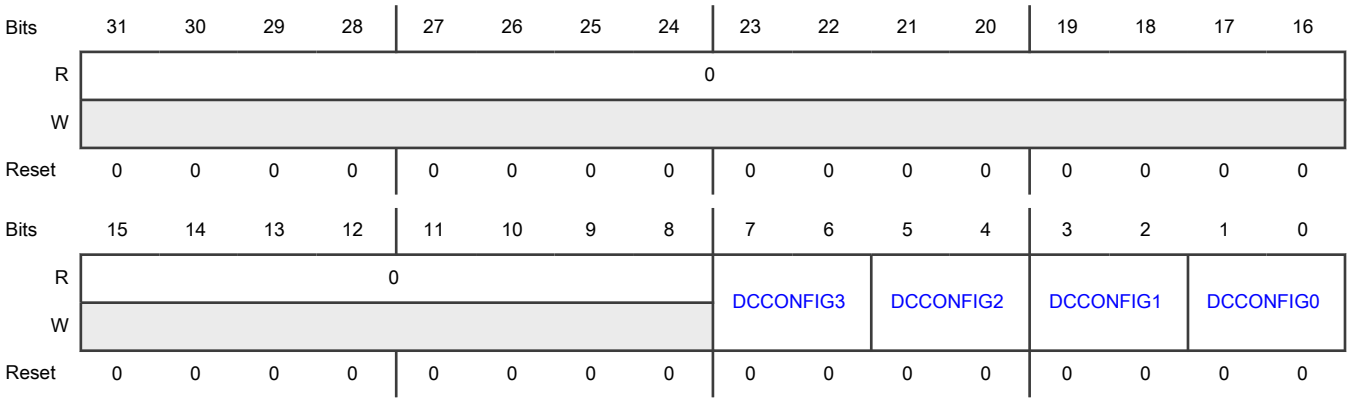
Offset

Register	Offset
DC_OUT_CTRL	68h

Function

Contains output DC remover configuration fields.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-6: DCCONFIG3 5-4: DCCONFIG2	Channel n DC Remover Configuration Defines the value of the cut-off frequency of the DC remover. 00b - 20 Hz (FS = 48 kHz)

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2: DCCONFIG1 1-0: DCCONFIG0	01b - 13.3 Hz (FS = 48 kHz) 10b - 40 Hz (FS = 48 kHz) 11b - DC remover is bypassed

76.7.10 MICFIL Range Control (RANGE_CTRL)

Offset

Register	Offset
RANGE_CTRL	74h

Function

Contains range configuration information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RANGEADJ3				RANGEADJ2				RANGEADJ1				RANGEADJ0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-12: RANGEADJ3 11-8: RANGEADJ2 7-4: RANGEADJ1	Channel n Range Adjustment Adjusts the dynamic range of the CIC filter. It is an unsigned positive value. You must configure this field within the following ranges: • If CTRL_2[QSEL] is in High-Quality and Very-Low-Quality 0 modes (HQ and VLQ0): [RANGEADJ*n*] ≤ 25 - ceil(5*$\log_2$(2OSR))

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0: RANGEADJ0	• If CTRL_2[QSEL] is in Medium-Quality and Very-Low Quality 1 modes (MQ and VLQ1): [RANGEADJ*n*] ≤ 25 - ceil(5*$\log_2$(OSR)) • If CTRL_2[QSEL] is in Low Quality and Very-Low-Quality 2 modes (LQ and VLQ2): [RANGEADJ*n*] ≤ 24 - ceil(5*$\log_2$(OSR)) where OSR = 16 - CICOSR, which is the CIC oversampling rate (see Equation 54), and the ceiling function, ceil(x), maps to the least integer greater than or equal to x.

76.7.11 MICFIL Range Status (RANGE_STAT)

Offset

Register	Offset
RANGE_STAT	7Ch

Function

Contains range status information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												RANGEUN...	RANGEUN...	RANGEUN...	RANGEUN...
W													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												RANGEOV...	RANGEOV...	RANGEOV...	RANGEOV...
W													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-16 RANGEUNFn	Channel n Range Underflow Error Flag Indicates an exceptional underflow condition in MICFIL range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE If this error flag is asserted, channel n data is not reliable. You must adjust the value of RANGE_CTRL[RANGEADJ*n*] until this flag is not asserted anymore. 0b - No exception by range underflow 1b - Exception by range underflow
15-4 —	Reserved
3-0 RANGEOVFn	Channel n Range Overflow Error Flag Indicates an exceptional overflow condition in MICFIL range. **NOTE** If this error flag is asserted, channel n data is not reliable. You must adjust the value of RANGE_CTRL[RANGEADJ*n*] until this flag is not asserted anymore. 0b - No exception by range overflow 1b - Exception by range overflow

76.7.12 Frame Synchronization Control (FSYNC_CTRL)

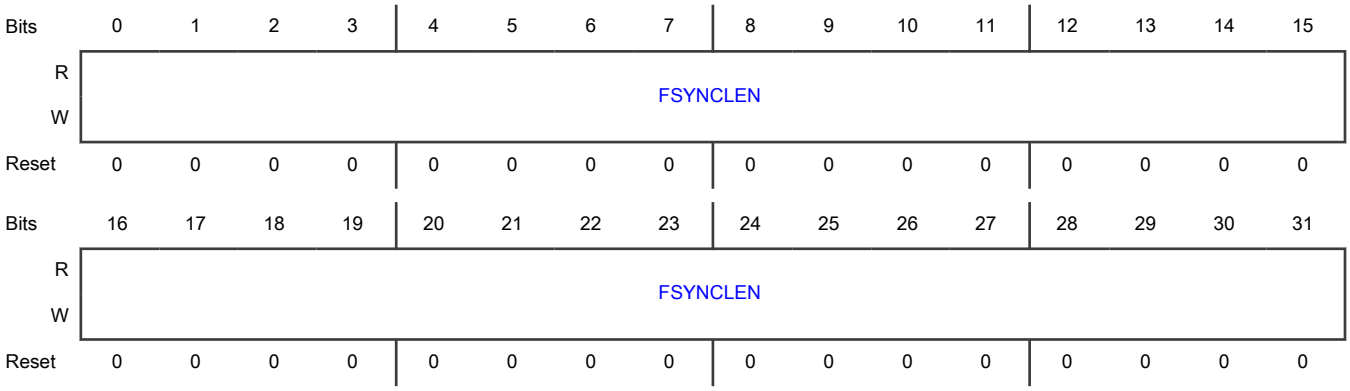
Offset

Register	Offset
FSYNC_CTRL	80h

Function

Contains the frame synchronization window length value.

Diagram



Fields

Field	Function
0-31 FSYNCLEN	Frame Synchronization Window Length Specifies the frame synchronization window length, which is equal to the value of FSYNC_CTRL[FSYNCLEN] + 1. See Frame synchronization for more information.

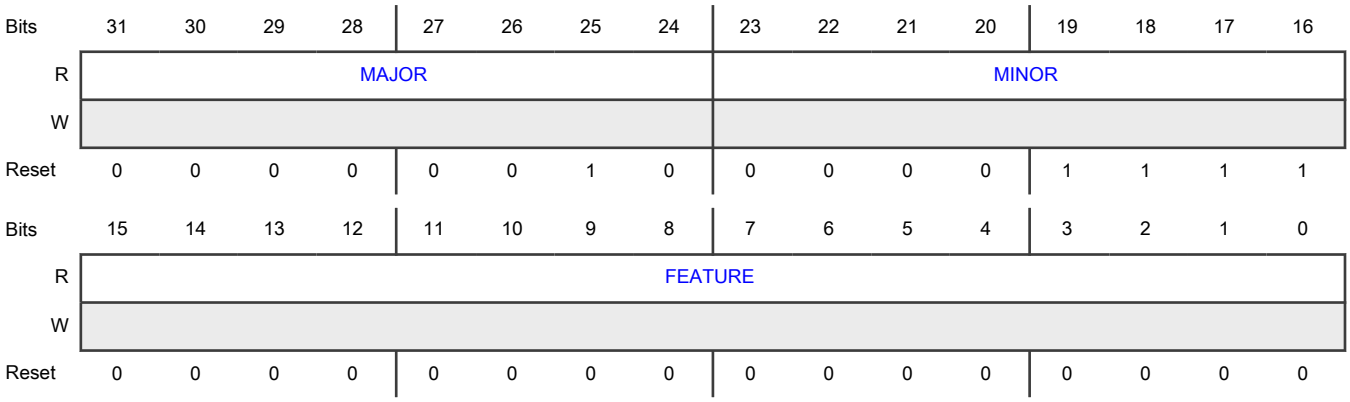
76.7.13 Version ID (VERID)

Offset

Register	Offset
VERID	84h

Function
Contains the major number, minor number, and feature specification number.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15-0 FEATURE	Feature Specification Number Returns the feature set number.

76.7.14 Parameter (PARAM)

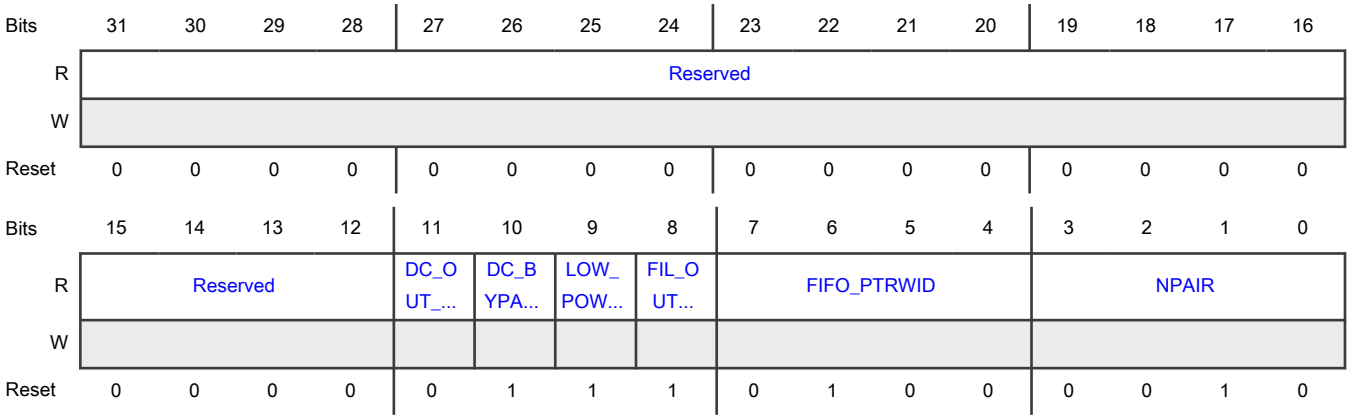
Offset

Register	Offset
PARAM	88h

Function

Contains MICFIL's general configuration information.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-12 —	Reserved
11 DC_OUT_BYPASS	Output DC Remover Bypass Indicates whether the DC remover at the output is permanently disabled (bypassed) in silicon. 0b - Active 1b - Disabled
10 DC_BYPASS	Input DC Remover Bypass Indicates whether the DC remover at the input is permanently disabled (bypassed) in silicon. 0b - Active 1b - Disabled
9	Low-Power Decimation Filter

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOW_POWER	0b - Disables 1b - Enables
8 FIL_OUT_WIDTH_24B	Filter Output Width 0b - 16 bits 1b - 24 bits
7-4 FIFO_PTRWID	FIFO Pointer Width 0000b - 0 bits 0001b - 1 bit 0010b - 2 bits 0011b-1110b - ... 1111b - 15 bits
3-0 NPAIR	Number of Microphone Pairs 0000b - None 0001b - 1 pair 0010b - 2 pairs 0011b-1110b - ... 1111b - 15 pairs

Chapter 77
Touch Sensing Input (TSI)

77.1 Chip-specific TSI information

Table 723. Reference links to related information

Topic	Related module	Reference
Full description	TSI	TSI
System memory map		Memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

77.1.1 Module instances

This device has one instance of the TSI module, TSI0.

77.1.2 Hardware triggers

LPTMR0 and LPTMR1 are used as the hardware triggers to TSI input source in this device. See TSI_TRIG register in INPUTMUX for details.

77.2 Overview

TSI provides touch sensing detection on capacitive touch sensors. The external capacitive touch sensor is typically formed on a printed circuit board (PCB) and the sensor electrodes are connected to TSI input channels through the I/O pins in the chip. The TSI module operates in switching-integration style to achieve low-power, high-sensitivity, and advanced electromagnetic compatibility (EMC) robustness.

The module supports:

- Self-Capacitance mode: In this mode, TSI requires only one pin for each touch sensor.
- Mutual-Capacitance mode: In this mode, TSI performs sensing using capacitive touch matrix in various TX-RX configurations. TSI requires one pin per TX and RX line.

TSI fully supports NXP touch library based on SDK, which provides a solid capacitive measurement module for the implementation of touch keyboard, rotaries, and sliders.

77.2.1 Block diagram

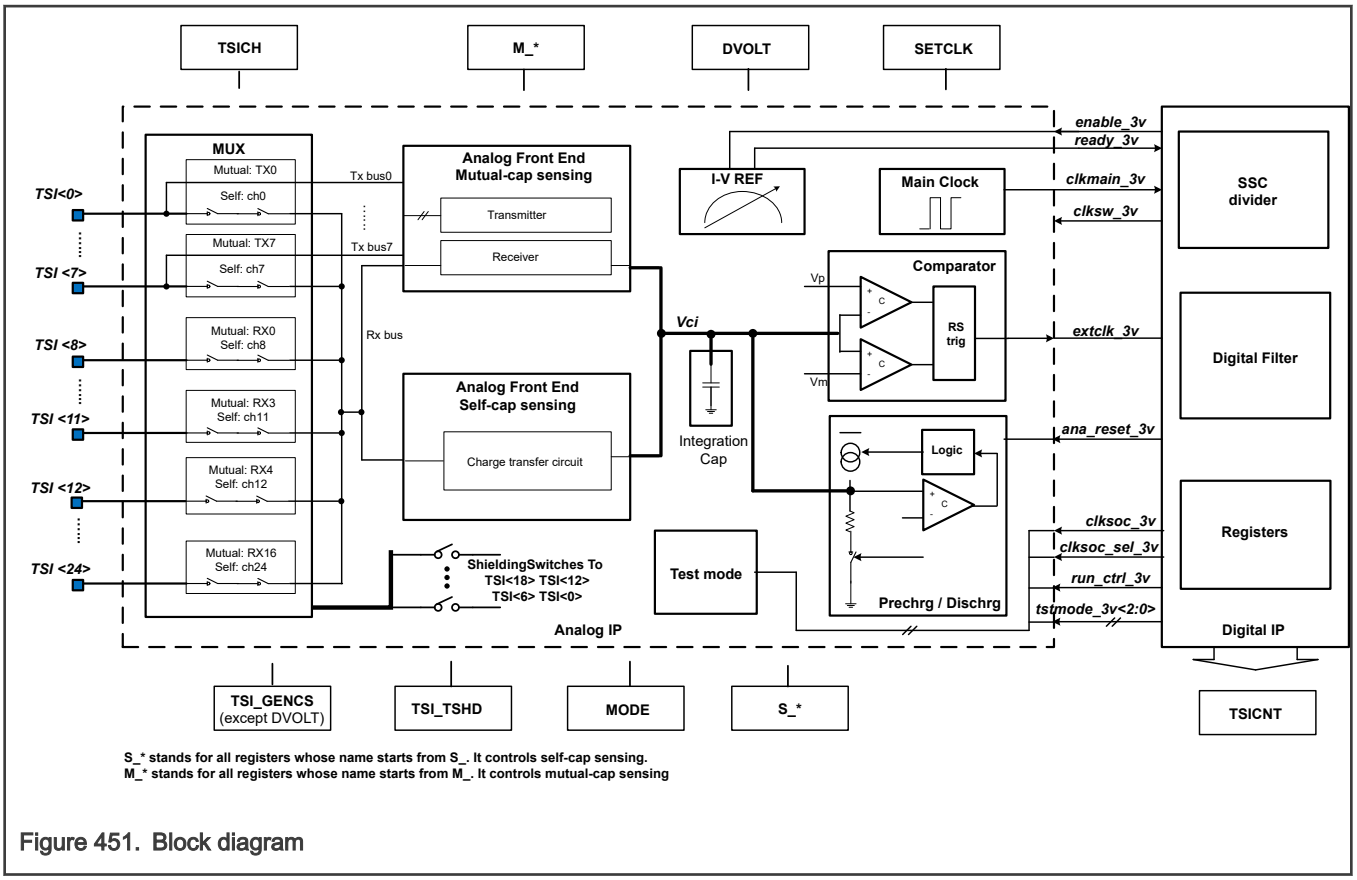


Figure 451. Block diagram

77.2.2 Features

- Achieves advanced EMC robustness
- Supports both Self-Capacitance and Mutual-Capacitance modes
- Requires one pin per electrode—no external components
- Provides adjustable touch sensing resolution and sensitivity for sensing various overlay materials and thicknesses
- Consumes low power
- Wakes up MCU from low-power modes for low-power application
- Supports DMA data transfer
- Supports NXP touch library that is based on SDK (see AN12709: NXP Touch Development Guide)

For electrode design recommendations, see AN3863: Designing Touch Sensing Electrodes.

77.3 Functional description

77.3.1 Touch sensor

77.3.1.1 Self-capacitance touch sensor

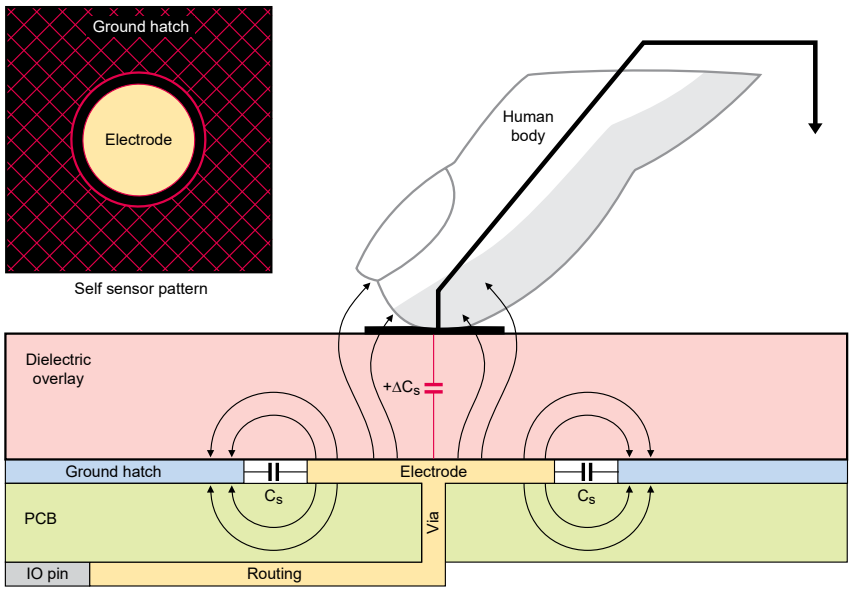


Figure 452. Self-capacitance touch sensor structure and electric field

Sensor structure:

- C_s is the intrinsic self-capacitance sensor (10 pF ~ 50 pF as usual).
- ΔC_s is the touch-generated self-capacitance sensor (0.3 pF ~ 2 pF as usual).
- $\Delta C_s \div C_s$ is the sensitivity of the sensor (1% ~ 10% as usual).

Intrinsic performance depends on the electrode pattern design, thickness, and dielectric of overlay, and PCB routing.

77.3.1.2 Mutual-capacitance touch sensor

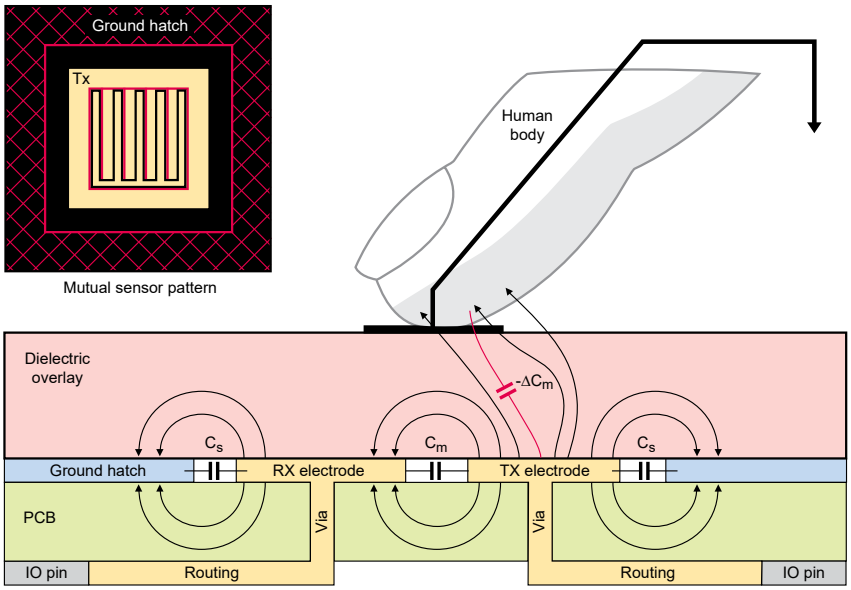


Figure 453. Mutual-capacitance touch sensor structure and electric field

Sensor structure:

- C_m is the intrinsic mutual-capacitance sensor (2 pF ~ 10 pF as usual).
- ΔC_m is the touch-reduced mutual-capacitance sensor (0.3 pF ~ 2 pF as usual).
- C_s is the parasitic self-capacitance sensor (10 pF ~ 50 pF as usual).
- $\Delta C_m \div C_m$ is the sensitivity of sensor (1% ~ 20% as usual).

Intrinsic performance depends on the electrode pattern design, thickness, and dielectric of overlay, and PCB routing.

77.3.2 Brief timing and operation of TSI

TSI operates by switching integration, whether under Self-Capacitance mode or Mutual-Capacitance mode.

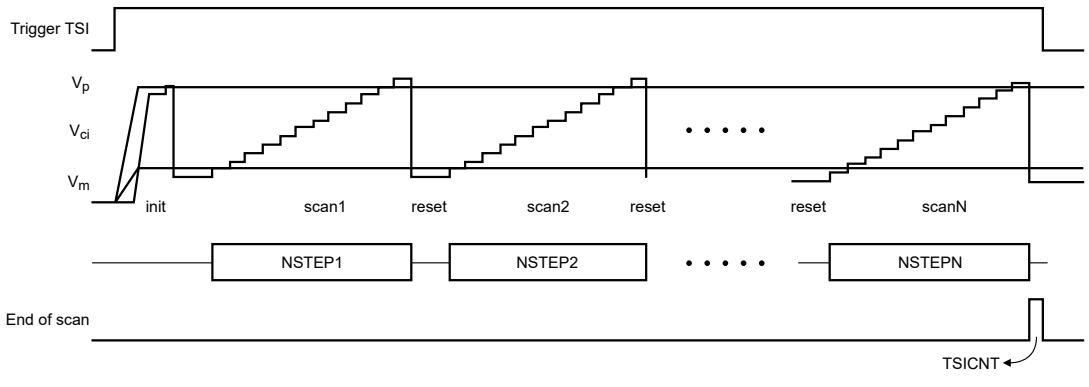


Figure 454. Brief timing of the TSI operation

$$TSICNT = NSTEP \times \frac{DECIMATION^{ORDER}}{CUTOFF}$$

$$SCANTIME = TNSTEP \times DECIMATION \times ORDER$$

Equation 55. Formula for calculating TSICNT and SCANTIME

where:

- DECIMATION specifies the times of scan, defined by the configuration of SINC[DECIMATION].
- ORDER specifies the order of sum up, defined by the configuration of SINC[ORDER].
- NSTEP specifies the analog integration steps that the module configuration and sensor define.
- TNSTEP is the scan time for getting each NSTEP.
- SCANTIME is the total scan time for getting the value of DATA[TSICNT].
- CUTOFF is the value of shifting out lower bits of the counter, defined by SINC[CUTOFF].

Operation

- TSI needs a very short initialization time for each trigger before it starts scanning the touch sensor (see Touch sensor for more).
- When scanning, the analog front end senses the self-capacitance or mutual-capacitance value and generates voltage steps on the integration capacitor. The step voltage depends on the touch sensor and module configuration.
- After the step voltage (V_{ci}) reaches threshold V_p of the comparator, the integration capacitor and analog front-end reset. The voltage V_{ci} discharges to V_m for next scanning.
- For each TSI trigger, you can set many scan times by using registers. Software uses the final count value, which is the summation of step numbers of each scan.
- The count relates to touch sensor capacitance (self-capacitance or mutual-capacitance) through formulas, and you can use it to sense touch event.

77.3.2.1 Self-capacitance sensing mode

Charge transfer operates through nonoverlapping clocks, ph1 and ph2, and a trans-conductance amplifier. The charge accumulates in integration capacitor C_i, which creates step voltage V_{ci}. See Equation 56 for NSTEP and TSINCT calculation.

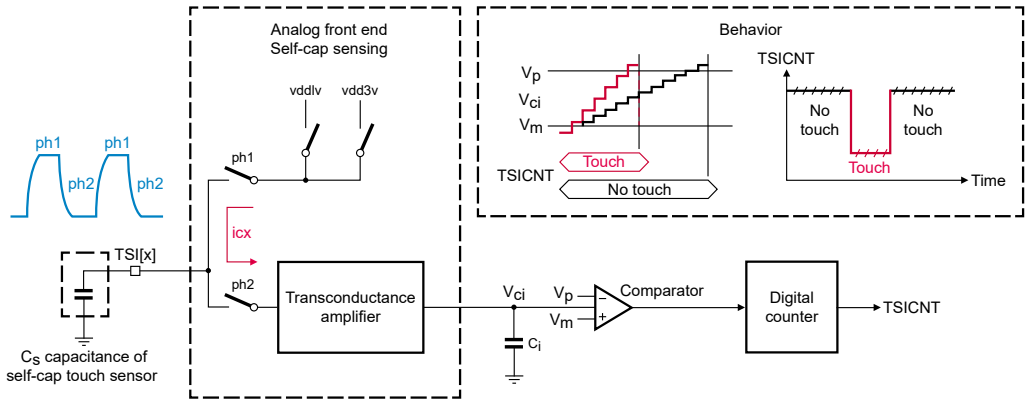


Figure 455. Self-capacitance sensing mode

$$NSTEP = \frac{C_i \times (V_p - V_m)}{vdd3v \times C_s \times S_XIN \times S_XCH}$$

$$TNSTEP = \frac{C_i \times (V_p - V_m)}{vdd3v \times C_s \times S_XIN \times S_XCH} \times \frac{1}{F_{sw}}$$

Equation 56. Formula to calculate NSTEP and TNSTEP

where:

- C_i is the integration capacitance (50 pF).
- V_p, V_m are dual reference voltages that you can configure using GENCS[DVOLT].
- vdd3v is the analog power supply voltage (3.3 V).
- S_XIN, and S_XCH are parameters for analog front-end that you can configure using CONFIG[S_XIN] and CONFIG[S_XCH], respectively.
- F_{sw} is the switching frequency that the SSC hardware module controls.
- C_s is the self-capacitance of touch sensor.

You can use GENCS[DVOLT], CONFIG[S_XIN], and CONFIG[S_XCH] to adjust the sensing resolution.

If the touch sensor intrinsic sensitivity is limited because of parasitic, you can write 1 to CONFIG[S_SEN] to activate the sensitivity boost feature. See Equation 57 to calculate NSTEP:

$$NSTEP = \frac{C_i \times (V_p - V_m)}{vdd3v \times \left(C_s - S_CTRIM \times \frac{S_XDN}{S_XCH}\right) \times S_XIN \times S_XCH}$$

Equation 57. NSTEP calculation

where:

- S_CTRIM is the internal trim capacitance, which you configure using CONFIG[S_CTRIM].
- S_XDN is a parameter of analog front-end, which you configure using CONFIG[S_XDN].

You can use CONFIG[S_CTRIM], CONFIG[S_XDN], and CONFIG[S_XCH] to adjust the sensitivity. The intrinsic sensitivity of the sensor is $\Delta C_s \div C_s$. See Equation 58 to calculate improved sensitivity:

$$\frac{\Delta C_S}{C_S - S_CTRIM \cdot \frac{S_XDN}{S_XCH}}$$

Equation 58. Improved sensitivity

If the touch sensor encounters a strong low-frequency noise, you can write 1 to CONFIG[S_NOISE] to activate noise cancelation. See Equation 59 to calculate NSTEP:

$$NSTEP = \frac{2C_i \times (V_p - V_m)}{(vdd3v - vddlv) \times C_s \times S_XIN \times S_XCH}$$

Equation 59. NSTEP calculation

where, vddlv is the internal pre-charge voltage (1.2 V).

During noise cancelation mode, vdd3v and vddlv are dual sample voltages. The analog front-end circuit samples the touch key twice, first using vdd3v (charging phase) and then using vddlv (discharging phase). At the end of each second phase, low-frequency noise is subtracted. In a long integration period, you can cancel noise-induced errors.

For example,

In a typical case, C_i = 50 pF, C_x = 25 pF, vdd3v = 3.3 V, DVOLT = 1 V, S_XIN = 1 ÷ 8, S_XCH = 1 ÷ 8; Dec = 8, Order = 2. Then, NSTEP = 38; TSICNT = 2432; SCANTIME = 0.62 ms.

NOTE

Do not write 1 to CONFIG[S_SEN] and CONFIG[S_NOISE] simultaneously.

77.3.2.2 Mutual-capacitance sensing mode

Mutual-capacitance sensing includes a transmitter and a receiver. Under clocking, the transmitter outputs pulses that couple through mutual-capacitance and then reach the receiver, which amplifies the signal and converts to charge current on integration capacitance C_i that creates step voltage V_{ci}.

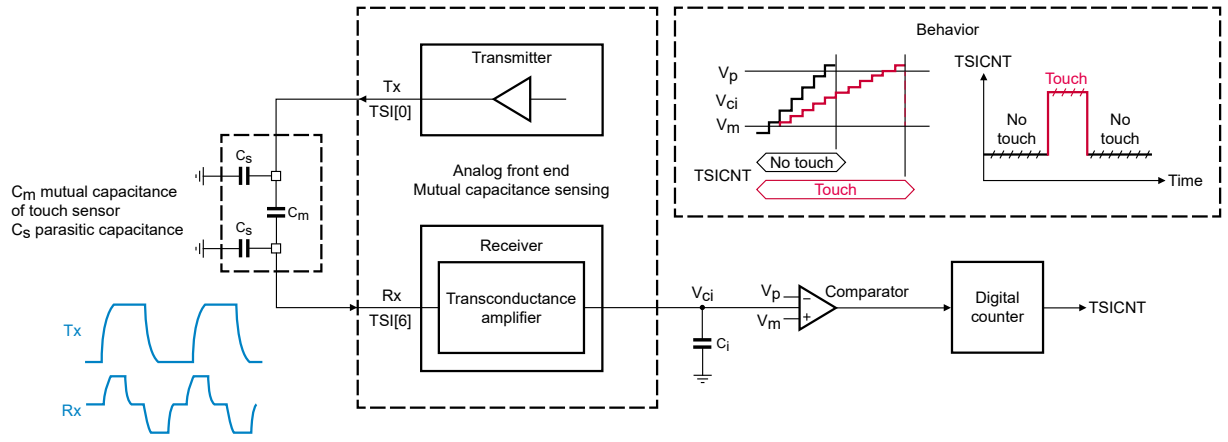


Figure 456. Mutual-capacitance sensing mode

Following are the formulas to calculate NSTEP and TNSTEP.

$$NSTEP = \frac{C_i \times (V_p - V_m) \times R_s}{\Delta V} \times \frac{M_PMIRRORL}{M_PMIRRORR} \times \frac{1}{t3}$$

Equation 60. NSTEP calculation

$$TNSTEP = \frac{C_i \times (V_p - V_m) \times R_s}{\Delta V} \times \frac{M_PMIRRORL}{M_PMIRRORR} \times \frac{T_{sw}}{t3}$$

Equation 61. TNSTEP calculation

where:

- C_i is the integration capacitance (50 pF).
- V_p, V_m are dual reference voltages that you can configure using GENCS[DVOLT].
- F_{sw} is the switching frequency that the spread spectrum clocking (SSC) hardware module controls (see SSC for more).
- T_{sw} is the switching period, and $T_{sw} = 1 \div F_{sw}$.
- t3 is the SSC output low period.
- R_s is a parameter for analog front-end that you can configure using SHIELD[M_SEN_RES].
- M is a parameter that CONFIG_MUTUAL[M_PMIRRORR] and CONFIG_MUTUAL[M_PMIRRORL] define.
- ΔV is signal voltage received.

$\Delta V = vdd3v \times C_m \div (C_m + C_s)$, which can be tens to hundreds of volts.

C_m, C_s are the mutual-capacitance and parasitic capacitance of the sensor.

If the touch sensor intrinsic sensitivity is limited because of parasitic, you can configure CONFIG_MUTUAL[M_SEN_BOOST] to activate the sensitivity boost feature. The basis average charge current is subtracted by boost current, which enlarges the signal current.

For example,

In a typical case, ΔV = 100 mV, R_s = 10 kΩ, $V_p - V_m$ = 1 V, C_i = 50 pF, M_PMIRRORL = 8, M_PMIRRORR = M_NMIRROR = 2, T_{sw} = 1 µs, t3 = 0.25 µs. NSTEP = 144, TNSTEP = 80 µs. Dec = 8, Order = 2, TSICNT = 80 × 64 = 5120, SCANTIME = 80 µs × 8 × 2 = 1280 µs.

NOTE

Let the configuration of CONFIG_MUTUAL[M_PMIRRORR] and CONFIG_MUTUAL[M_NMIRROR] be the same.

77.3.3 Enable TSI

TSI is fully functional in Run, Wait, and Low-Power modes. You must write 1 to GENCS[TSIEN] to enable TSI in Run and Wait modes. When GENCS[STPE] = 1, it allows TSI to work in Low-Power mode.

77.3.4 Wake-up MCU from low-power modes

In low-power modes, when enabled by GENCS[STPE], TSI can bring MCU out of its low-power modes (STOP, VLPS, and so on) by using either the end-of-scan or out-of-range interrupt. This means, if GENCS[ESOR] = 1, the end-of-scan interrupt is selected. Otherwise, the out-of-range interrupt is selected.

77.3.5 Software and hardware triggers

TSI allows a software or hardware trigger to start a scan. When you apply a software trigger (GENCS[STM] = 0), you must write 1 to GENCS[SWTS] to start the scan electrode channel that CONFIG[TSICH] identifies. When a hardware trigger is applied (GENCS[STM] = 1), TSI does not start scanning until the hardware trigger arrives. The hardware trigger differs depending on the MCU configuration. Generally, it could be an event that RTC overflows. See the chip configuration section for more information.

77.3.6 Scan times

TSI provides a multi-scan function. SINC[DECIMATION] indicates the number of scans that allow the scan number to be configured from 1 to 32. If you write 0 to SINC[DECIMATION] (only once), a single scan is engaged. The 16-bit counter accumulates all scan results until the number of scan times reaches the value of SINC[DECIMATION], and you can read DATA[TSICNT] to get this accumulation. When you enable DMA transfer, the DMA engine can also read out the counter values.

77.3.7 Reference voltage

You must use a reference voltage to set up the rampup threshold. It determines TSICNT and SCANTIME. TSI offers dual reference voltages for both comparators. The internal reference voltage can work in low-power modes even when the MCU regulator is partially powered down, which is ideally for low-power touch detection.

You can configure reference voltage by using GENCS[DVOLT]. The following table shows all delta voltage configurations.

Table 724. Delta voltage configuration

DVOLT	V_m (V)	V_p (V)	ΔV (V)
000	0.6	1.7	1.1
001	0.6	1.9	1.3
010	0.6	2.1	1.5
011	0.6	2.3	1.7
100	0.6	2.5	1.9
101	0.6	2.7	2.1

77.3.8 End of scan

After the scan completes, DATA[EOSF] becomes 1. If GENCS[ESOR] = 1 and GENCS[DMAEN_EOS] = 0, an interrupt is submitted to CPU for post-processing immediately. You can also use the interrupt to wake up the MCU to execute ISR if it is in a low-power mode. You can enable the DMA function by writing 1 to GENCS[DMAEN_EOS]. As soon as the scan completes, a DMA transfer request is asserted to the DMA controller for data movement. Generally, the DMA engine fetches TSI conversion result from TSI Data and Status (DATA), stores it in another memory space, and then refreshes the TSI scan channel index (CONFIG[TSICH]) for the next loop. After a DMA transfer completes, DATA[EOSF] becomes 0 automatically.

77.3.9 Modes

Table 725. Modes

Mode	Description
Stop and Low-Power Stop	TSI is fully functional in all stop modes if GENCS[STPE] = 1. CONFIG[TSICH] specifies the channel for scanning upon the trigger. After a scan completes, you can select either the end-of-scan or out-of-range interrupt to bring MCU out of low-power modes.
Wait	TSI is fully functional in this mode. After a scan completes, TSI submits an interrupt request to CPU if the interrupt is enabled.
Run	TSI is fully functional in this mode. After a scan completes, TSI submits an interrupt request to CPU if the interrupt is enabled.

77.3.10 Clocking

Switching clock with frequency F_{sw} drives both self-capacitance front-end and mutual-capacitance front-end. It is derived from the SSC clock with flatten emission energy. In addition, TSI SSC 0 (SSC0), TSI SSC 1 (SSC1), and TSI SSC 2 (SSC2) configure the frequency of switching clock (see SSC for more information). The SSC clock source is the main clock module in TSI. You can configure the frequency of the main clock by using GENCS[SETCLK].

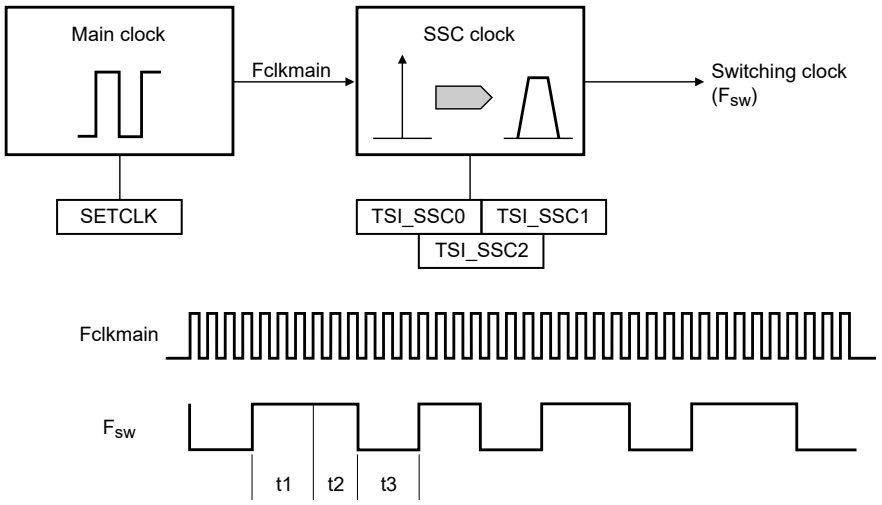


Figure 457. Clocking

For example,

- To use a non-SSC switching clock with a frequency of 1 MHz:
  1. Write 011 to GENCS[SETCLK] for Fclkmain = 16.65 MHz.
  2. Write 10 to SSC0[SSC_MODE] to disable the SSC function.
  3. Write 0000–0111 to SSC0[SSC_PRESCALE_NUM] for Fclkmain ÷ 8. If SSC mode is disabled, the frequency formula is Fclkmain ÷ [(SSC_PRESCALE_NUM + 1) × 2].

Let TSI SSC 0 (SSC0), TSI SSC 1 (SSC1), and TSI SSC 2 (SSC2) retain their default values. Then, F_{sw} = 16.65 MHz ÷ (8 × 2) = 1.04 MHz. F_{sw} is square wave pulse.

- To use Pseudo-Random Binary Sequence (PRBS) mode having SSC switching clock with a central frequency of 1 MHz:
  1. Write 011 to GENCS[SETCLK] for Fclkmain = 16.65 MHz. Then, the period of main clock is Tclkmain.
  2. Write 00 to SSC0[SSC_MODE] to enable PRBS SSC mode.
  3. Write 0100 to SSC0[BASE_NOCHARGE_NUM] to set t1 = 5 × Tclkmain × (SSC_PRESCALE_NUM + 1).
  4. Write 0110 to SSC0[CHARGE_NUM] to set t3 = 7 × Tclkmain × (SSC_PRESCALE_NUM + 1).
  5. Write 0110 to SSC0[PRBS_OUTSEL] to set t2 range from 1 × Tclkmain to 6 × Tclkmain. The average t2 is 3.5 × Tclkmain × (SSC_PRESCALE_NUM + 1).

Let TSI SSC 0 (SSC0), TSI SSC 1 (SSC1), and TSI SSC 2 (SSC2) retain their default values. Then, F_{sw} = 16.65 MHz ÷ [(5 + 3.5 + 7) × (0 + 1)] = 1.074 MHz. F_{sw} is the spectrum spread pulse.

77.3.10.1 SSC

In capacitance touch sense systems, the baseband signal is narrowband and nearing the DC, and the noise is a wideband noise. For lower cost, the cut-off frequency of the antialiasing low-pass filter at analog front-end is not low enough compared to the sampling frequency. Therefore, when sampling, the noises for which the frequency is nearing sampling frequency are overlapped to baseband. For solving this problem in capacitance touch sense systems, you can involve a low-cost SSC. The SSC center frequency and frequency span range must be flexible enough for handling various frequency noises.

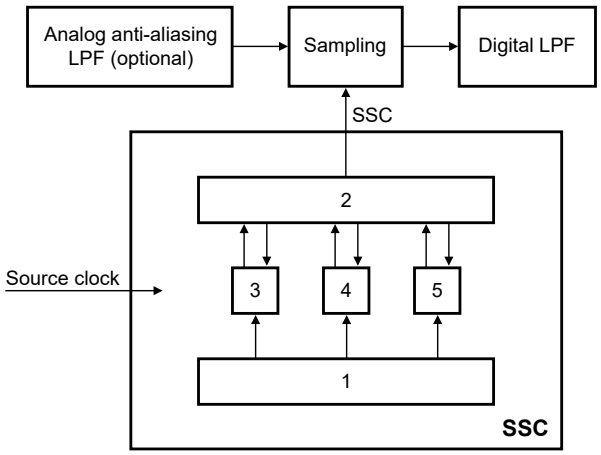


Figure 458. SSC

With this SSC, the noises for which the frequency is nearing sampling frequency can be spanned to a wider frequency range instead of a single-peak frequency noise, and only parts of the noise overlap into baseband (because baseband is a narrowband). This promotes signal-to-noise ratio (SNR).

SSC comprises five components that work together and generate a configurable center frequency and span frequency range:

- Configurable registers, generating all configurable settings for components 3, 4, and 5, by using TSI SSC 0 (SSC0), TSI SSC 1 (SSC1), and TSI SSC 2 (SSC2).
- State machine engine, controlling and monitoring components 3, 4, and 5.
- A configurable counter to generate logic 1 in the SSC output clock; SSC0[BASE_NOCHARGE_NUM] controls the max value of the counter.
- A configurable up-down counter or a PRBS method to generate logic 1 in the SSC output clock. When using the up-down counter, SSC2[MOVE_NOCHARGE_MIN] and SSC2[MOVE_NOCHARGE_MAX] limit the counter value. When using the PRBS method, the output of the PRBS method controls the length of logic 1.
- A configurable up-down counter to generate logic 0 in the SSC output clock; SSC0[CHARGE_NUM] controls the max value of the counter.

Figure 459 illustrates the timing of the input system clock and SSC output bit.

- t1: Component 3 and SSC0[BASE_NOCHARGE_NUM] control it.
- t2: When using the up-down counter, component 4, SSC2[MOVE_NOCHARGE_MIN], and SSC2[MOVE_NOCHARGE_MAX] control it. When using the PRBS method, the PRBS output controls it.
- t3: Component 5 and SSC0[CHARGE_NUM] control it.

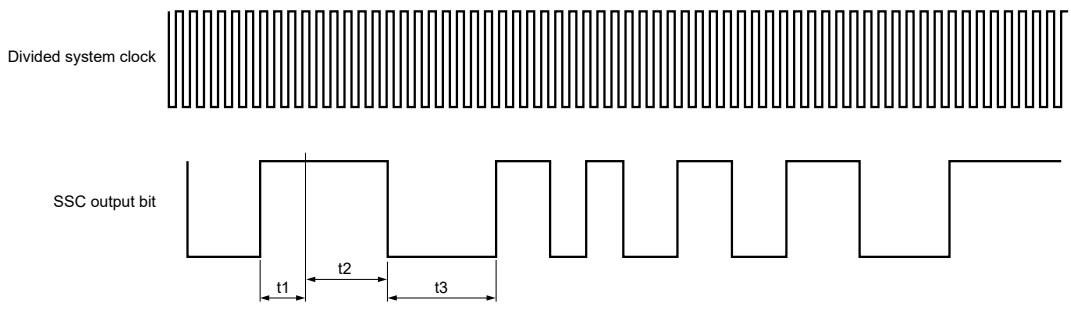


Figure 459. Timing of SSC

Equation 62 defines the average frequency of the SSC output.

$$\text{frequency}_{SSC} = \frac{\text{frequency}_{system}}{(t1 + t2 + t3) \times (SSC_PRESCALE_NUM + 1)}$$

Equation 62. SSC average frequency

Figure 460 shows the linear feedback shift register (LFSR) circuit that is used for generating the SSC output bit by using the PRBS method.

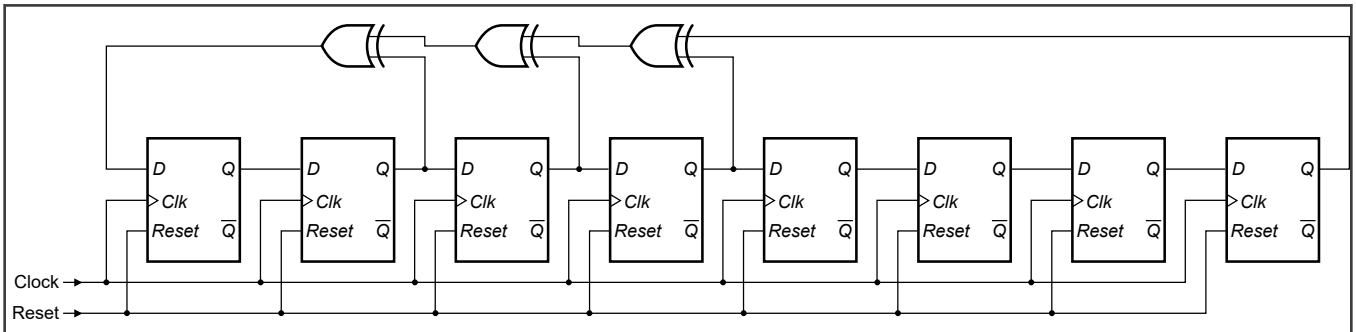


Figure 460. LFSR circuit

77.3.11 Reset

There is only one reset on this module, ipg_hard_async_reset_b. After it becomes 0, all TSI registers reset to their default values.

77.3.12 Interrupts

If enabled, TSI scans the electrode that CONFIG[TSICH] specifies, as soon as the trigger arrives. DATA[OUTRGF] generates a TSI interrupt request if GENCS[OUTRG_EN] becomes 0. With this configuration, after the end-of-electrode scan, the electrode capacitance is converted and stored in DATA[TSICNT]. The out-of-range interrupt is only requested if TSI Threshold (TSHD) defines a considerable capacitance change. For instance, when in a low-power mode, the electrode capacitance does not vary, and the out-of-range interrupt does not interrupt the CPU. You can use TSI Threshold (TSHD) for out-of-range interrupt generation.

77.3.13 DMA

DMA supports transmit only when GENCS[DMAEN_OUTRG] or GENCS[DMAEN_EOS] = 1. A DMA transfer request is asserted when DATA[EOSF] or DATA[OUTRGF] is set. Then the on-chip DMA controller detects this request and transfers data between the memory space and TSI register space. After data transfer, DMA is asserted to clear DATA[EOSF] or DATA[OUTRGF] automatically. The DMA controller uses this function normally to get the conversion result from DATA[TSICNT] upon an end-of-scan or out-of-range event and then refreshes the channel index (DATA[TSICNT]) for the next trigger.

77.4 External signals

TSI contains up to 25 external pins for touch sensing. The following tables describe each of the TSI external pins.

Table 726. Signal description (self-capacitance sensing)

Name	Port	Direction	Function	Reset state
TSI[24:0]	TSI	I/O	TSI sensing pins or GPIO pins	I/O

Table 727. Signal description (mutual-capacitance sensing)

Name	Port	Direction	Function	Reset state
TSI[7:0]	TSI	I/O	TSI TX pins or GPIO pins	I/O
TSI[24:8]	TSI	I/O	TSI RX pins or GPIO pins	I/O

77.4.1 TSI[24:0]

When the TSI functionality is enabled, the TSI analog portion uses the corresponding channel to connect external onboard touch capacitors. The PCB connection between the pin and touch pad must be as short as possible to reduce parasitic capacity on board.

77.5 Initialization

This module does not require initialization.

77.6 Application information

To enable TSI:

1. Specify all the registers needed to pick up the functions to use (for example, channel number, scan mode, decimation, clock setting, V_p, V_m value, and SSC mode).
2. Write 1 to GENCS[TSIEN].
3. Send a software or hardware trigger to the TSI module. See Software and hardware triggers for more information.
4. Read the scan counter value after DATA[EOSF] becomes 1.

To disable TSI, write 0 to GENCS[TSIEN].

77.7 Register definition

77.7.1 TSI register descriptions

77.7.1.1 TSI memory map

TSI0 base address: 4005_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	TSI CONFIG (TSI_CONFIG) for Self-Capacitor (CONFIG)	32	RW	0780_0000h
0h	TSI CONFIG (TSI_CONFIG) for Mutual-Capacitor (CONFIG_MUTUAL)	32	RW	0780_0000h
4h	TSI Threshold (TSHD)	32	RW	0000_0000h
8h	TSI General Control and Status (GENCS)	32	RW	0200_1000h
Ch	TSI Mutual-Capacitance (MUL)	32	RW	0000_0000h
10h	TSI SINC Filter (SINC)	32	RW	0007_0001h
14h	TSI SSC 0 (SSC0)	32	RW	6032_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
18h	TSI SSC 1 (SSC1)	32	RW	0060_0040h
1Ch	TSI SSC 2 (SSC2)	32	RW	1008_0101h
20h	TSI Baseline (BASELINE)	32	RW	0001_0000h
24h	TSI Channel Merge (CHMERGE)	32	RW	0000_0000h
28h	TSI Shield (SHIELD)	32	RW	0400_0000h
100h	TSI Data and Status (DATA)	32	RW	0000_0000h
108h	TSI Miscellaneous (MISC)	32	RW	0000_0000h
10Ch	TSI AUTO TRIG (TRIG)	32	RW	0000_0000h

77.7.1.2 TSI CONFIG (TSI_CONFIG) for Self-Capacitor (CONFIG)

Offset

Register	Offset
CONFIG	0h

Function

Provides self- and mutual-capacitor control and configuration information for TSI. Two configuration registers share one address.

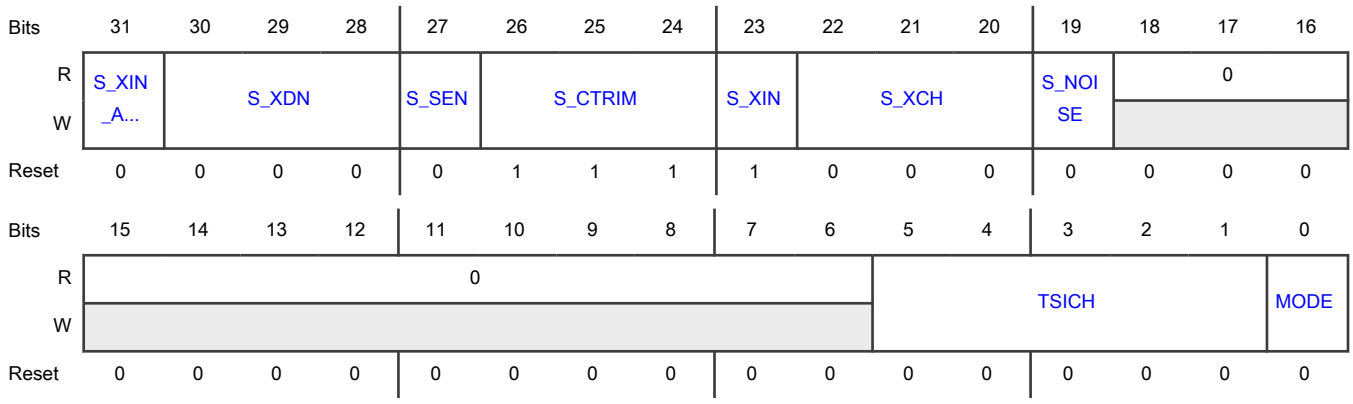
NOTE

As CONFIG and CONFIG_MUTUAL registers share the same offset, CONFIG[MODE] controls their reset value. After a system reset occurs, when CONFIG[MODE] becomes 1, the reset values of CONFIG and CONFIG_MUTUAL registers automatically change to 4000001h.

NOTE

When CONFIG[MODE] = 0, this register provides self-capacitor control and configuration information for TSI.

Diagram



Fields

Field	Function
31 S_XIN_ADD	S_XIN Adjust Ratio Disables the S_XIN adjust ratio. 0b - Disables; S_XIN = 0 for 1 ÷ 4, S_XIN = 1 for 1 ÷ 8 1b - Enables; S_XIN = 0 for 1 ÷ 8, S_XIN = 1 for 1 ÷ 16
30-28 S_XDN	Self-Capacitance Discharge Current Multiple Specifies the charge or discharge current multiple for sensitivity boost. When CONFIG[S_SEN] = 1, you are able to adjust sensitivity. 000b - 1 ÷ 16 001b - 1 ÷ 8 010b - 1 ÷ 4 011b - 1 ÷ 2
27 S_SEN	Self-Capacitance Sensitivity Boost Disables sensitivity boost mode for self capacitance. 0b - Disables 1b - Enables
26-24 S_CTRIM	Capacitor Trim Setting Specifies capacitor trim setting. When CONFIG[S_SEN] = 1, you are able to adjust sensitivity. 000b - 2.5 pF 001b - 5.0 pF 010b - 7.5 pF 011b - 10 pF 100b - 12.5 pF 101b - 15.0 pF 110b - 17.5 pF 111b - 20 pF
23 S_XIN	Self-Capacitance Input Current Multiple Specifies the input current multiple. 0b - 1 ÷ 8 1b - 1 ÷ 4
22-20 S_XCH	Self-Capacitance Charge Current Multiple Specifies the charge or discharge current multiple for charge transfer. 000b - 1 ÷ 16

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001b - 1 ÷ 8 010b - 1 ÷ 4 011b - 1 ÷ 2
19 S_NOISE	Self-Capacitance Noise Cancelation Disables noise cancelation mode for self capacitance. 0b - Disables 1b - Enables
18-6 —	Reserved
5-1 TSICH	TSI Channel Specifies the current channel to be measured for Self-Capacitance mode. In Hardware Trigger mode, (GENCS[STM] = 1), the scan does not start until the hardware trigger occurs. In Software Trigger mode, (GENCS[STM] = 0), the scan starts immediately when you write 1 to GENCS[SWTS]. 0_0000b - Channel 0 0_0001b - Channel 1 0_0010b - Channel 2 0_0011b - Channel 3 0_0100b - Channel 4 0_0101b - Channel 5 0_0110b - Channel 6 0_0111b - Channel 7 0_1000b - Channel 8 0_1001b - Channel 9 0_1010b - Channel 10 0_1011b - Channel 11 0_1100b - Channel 12 0_1101b - Channel 13 0_1110b - Channel 14 0_1111b - Channel 15 1_0000b - Channel 16 1_0001b - Channel 17 1_0010b - Channel 18 1_0011b - Channel 19

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1_0100b - Channel 20 1_0101b - Channel 21 1_0110b - Channel 22 1_0111b - Channel 23 1_1000b - Channel 24
0 MODE	Mode Specifies the sensing mode used for self capacitance and mutual capacitance. For both the modes, the address is TSI_CONFIG. 0b - Self capacitance 1b - Mutual capacitance

77.7.1.3 TSI CONFIG (TSI_CONFIG) for Mutual-Capacitor (CONFIG_MUTUAL)

Offset

Register	Offset
CONFIG_MUTUAL	0h

Function

Provides self- and mutual-capacitor control and configuration information for TSI. Two configuration registers share one address.

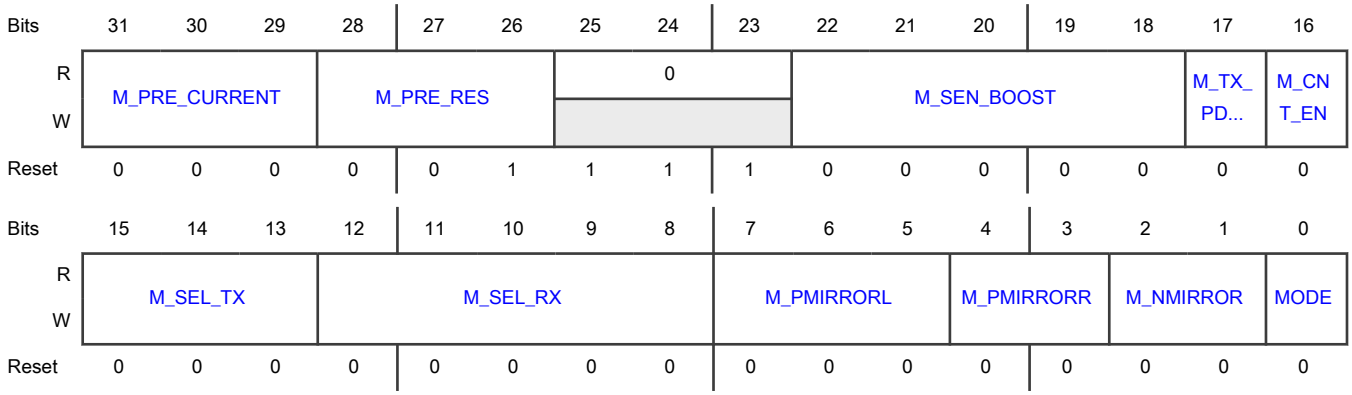
NOTE

As CONFIG and CONFIG_MUTUAL registers share the same offset, CONFIG[MODE] controls their reset value. After a system reset occurs, when CONFIG[MODE] becomes 1, the reset values of CONFIG and CONFIG_MUTUAL registers automatically change to 4000001h.

NOTE

When CONFIG[MODE] = 1, this register provides mutual-capacitor control and configuration information for TSI.

Diagram



Fields

Field	Function
31-29 M_PRE_CURRENT	Mutual-Capacitance Precharge Current Specifies the current used in Vref generator. 000b - 1 µA 001b - 2 µA 010b - 3 µA 011b - 4 µA 100b - 5 µA 101b - 6 µA 110b - 7 µA 111b - 8 µA
28-26 M_PRE_RES	Mutual-Capacitance Precharge Resistor Specifies the resistor used in precharge. 000b - 1 kΩ 001b - 2 kΩ 010b - 3 kΩ 011b - 4 kΩ 100b - 5 kΩ 101b - 6 kΩ 110b - 7 kΩ 111b - 8 kΩ
25-23 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
22-18 M_SEN_BOOST	Mutual-Capacitance Sensitivity Boost Specifies the sensitivity boost current. 0_0000b - 0 µA 0_0001b - 2 µA 0_0010b - 4 µA 0_0011b - 6 µA 0_0100b - 8 µA 0_0101b - 10 µA 0_0110b - 12 µA 0_0111b - 14 µA 1_1111b - 62 µA 1_xxxxb - 2 × n µA
17 M_TX_PD_EN	Mutual-Capacitance TX Pulldown Enable Enables the unused TX pulldown. In Mutual-Capacitance mode, if this field = 1, all unused TX are pulled down to ground. 0b - Disables 1b - Enables
16 M_CNT_EN	Mutual-Capacitance Counter Enable Enables the counter signal for Mutual-Capacitance mode. If this field = 0, the counter works throughout the scan process. If this field = 1, the counter only works after the capacitor has performed the integration. 0b - Disables 1b - Enables
15-13 M_SEL_TX	Mutual-Capacitance TX Channel Selection When CONFIG_MUTUAL[MODE] = 1, selects the TX channel. 000b - TSI[0] 001b - TSI[1] 010b - TSI[2] 011b - TSI[3] 100b - TSI[4] 101b - TSI[5] 110b - TSI[6] 111b - TSI[7]

Table continues on the next page...

Table continued from the previous page...

Field	Function
12-8 M_SEL_RX	Mutual-Capacitance RX Channel Selection When CONFIG_MUTUAL[MODE] = 1, selects the RX channel. 0_0000b - TSI[8] 0_0001b - TSI[9] 0_0010b - TSI[10] 0_0011b - TSI[11] 0_0100b - TSI[12] 0_0101b - TSI[13] 0_0110b - TSI[14] 0_0111b - TSI[15] 0_1000b - TSI[16] 0_1001b - TSI[17] 0_1010b - TSI[18] 0_1011b - TSI[19] 0_1100b - TSI[20] 0_1101b - TSI[21] 0_1110b - TSI[22] 0_1111b - TSI[23] 1_0000b - TSI[24]
7-5 M_PMIRRORL	PMOS Current Mirror on Left Side Specifies the PMOS current mirror on the left side. 000b - m = 4 001b - m = 8 010b - m = 12 011b - m = 16 100b - m = 20 101b - m = 24 110b - m = 28 111b - m = 32
4-3 M_PMIRRORR	PMOS Current Mirror on Right Side Specifies the PMOS current mirror on the right side. 00b - m = 1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - m = 2 10b - m = 3 11b - m = 4
2-1 M_NMIRROR	NMOS Current Mirror Specifies the NMOS current mirror. 00b - m = 1 01b - m = 2 10b - m = 3 11b - m = 4
0 MODE	Mode Specifies the sensing mode for self capacitance and mutual capacitance. For both, the address is TSI_CONFIG. 0b - Self capacitance 1b - Mutual capacitance

77.7.1.4 TSI Threshold (TSHD)

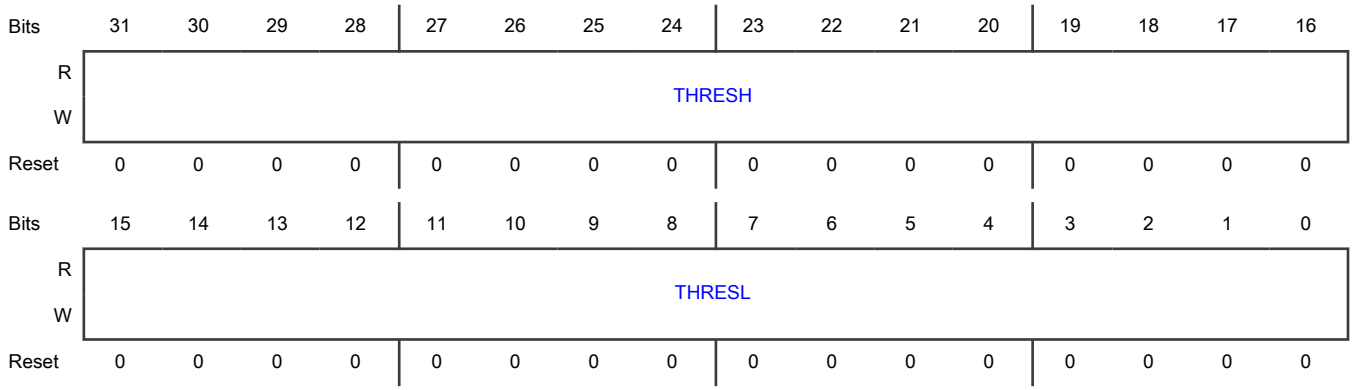
Offset

Register	Offset
TSHD	4h

Function

Defines the high and low thresholds of the count value.

Diagram



Fields

Field	Function
31-16 THRESH	TSI Wakeup Channel High Threshold Specifies the high threshold of the wakeup channel. The threshold values can be used as low-power wakeup thresholds.
15-0 THRESL	TSI Wakeup Channel Low Threshold Specifies the low threshold of the wakeup channel. The threshold values can be used as low-power wakeup thresholds.

77.7.1.5 TSI General Control and Status (GENCS)

Offset

Register	Offset
GENCS	8h

Function

Provides control and configuration information for TSI.

NOTE

When TSI is working, you must not change the configuration fields, GENCS[TSIEN] and GENCS[STM].

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	OUTR	0		ESOR	SETCLK			0	S_PR	0	DEBOUNCE				
W		G_EN								OX_...						
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	DVOLT			CTRIM_FINE			0	SWTS	0	TSIEN	STPE	STM	DMAE	0	DMAE
W														N_O...		N_E...
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30	Out-of-Range Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
OUTRG_EN	Enables the out-of-range interrupt. 0b - Disables 1b - Enables
29-28 —	Reserved
27 ESOR	End-of-Scan Interrupt Enable Enables the end-of-scan interrupt. 0b - Disables 1b - Enables
26-24 SETCLK	Set Clock Sets the main clock frequency. 000b - 27.37 MHz 001b - 22.23 MHz 010b - 18.73 MHz 011b - 16.65 MHz 100b - 14.27 MHz 101b - 12.73 MHz 110b - 11.49 MHz 111b - 10.46 MHz
23 —	Reserved
22 S_PROX_EN	Proximity Enable Signal Enables the proximity function. 0b - Disables 1b - Enables
21 —	Reserved
20-16 DEBOUNCE	Debounce Sets the debounce function by specifying the number of out-of-range events that can generate an interrupt. 0_0000b - 1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0001b - 2 1_1111b - 31 1_xxxxb - n
15 —	Reserved
14-12 DVOLT	Delta Voltage Selects the comparator V_m, V_p. 000b - Vm = 0.6 V, Vp = 1.7 V 001b - Vm = 0.6 V, Vp = 1.9 V 010b - Vm = 0.6 V, Vp = 2.1 V 011b - Vm = 0.6 V, Vp = 2.3 V 100b - Vm = 0.6 V, Vp = 2.5 V 101b - Vm = 0.6 V, Vp = 2.7 V
11-9 CTRIM_FINE	Capacitor Fine Trim Specifies additional trim bits for sense boost adjust. 000b - 0.3125 pF 001b - 0.625 pF 010b - 0.3125 × 3 pF 011b - 0.3125 × 4 pF 100b - 0.3125 × 5 pF 101b - 0.3125 × 6 pF 110b - 2.1875 pF 111b - 2.5 pF
8 —	Reserved
7 SWTS	Software Trigger Start Specifies whether software trigger start takes effect. If GENCS[STM] = 0, writing 1 to this field starts a scan. Read value is always 0. 0b - No effect 1b - Takes effect
6 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 TSIEN	TSI Enable Enables TSI. 0b - Disables 1b - Enables
4 STPE	TSI Stop Enable Enables TSI to function in low-power modes (stop, VLPS, and so on). If this field = 0, MCU goes into a low-power mode. If this field = 1, TSI continues to run in all low-power modes. 0b - Disables 1b - Enables
3 STM	Scan Trigger Mode Specifies the trigger mode used. You are allowed to change this field when TSI is not working. 0b - Software trigger scan 1b - Hardware trigger scan
2 DMAEN_OUTRG	Out-of-Range DMA Transfer Request Enable Enables the DMA request at the out-of-range event. 0b - Disables 1b - Enables
1 —	Reserved
0 DMAEN_EOS	In-Progress DMA Transfer Request Enable Enables the DMA request at the end-of-scan event. 0b - Disables 1b - Enables

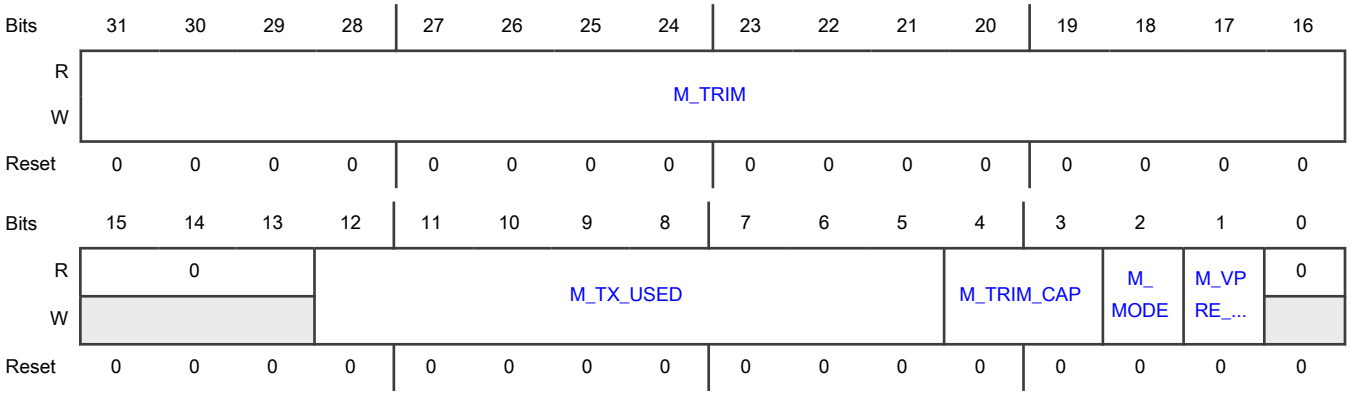
77.7.1.6 TSI Mutual-Capacitance (MUL)

Offset

Register	Offset
MUL	Ch

Function
Provides control for Mutual-Capacitance mode.

Diagram



Fields

Field	Function
31-16 M_TRIM	Mutual-Capacitance Trim Specifies the mutual-capacitance trim function.
15-13 —	Reserved
12-5 M_TX_USED	Mutual-Capacitance TX Used Indicates the channel that is used for mutual-capacitance TX when CONFIG[MODE] = 1. 0000_0000b - GPIO 0000_0001b - Mutual capacitance
4-3 M_TRIM_CAP	Mutual-Capacitance Trim Cap Adjusts Cp in Mutual-Capacitance mode. 00b - 0 pF 01b - 10 pF 10b - 10 pF 11b - 20 pF
2 M_MODE	Mutual-Capacitance Mode Controls TX drive mode. 0b - – 5 V ~ + 5 V 1b - 0 V ~ + 5 V
1 M_VPRE_CHOOSE	Mutual-Capacitance Prevoltage Specifies the digital control signal for pre-voltage selection. 0b - Internal 1.2 V 1b - External 1.2 V from PMC

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 —	Reserved

77.7.1.7 TSI SINC Filter (SINC)

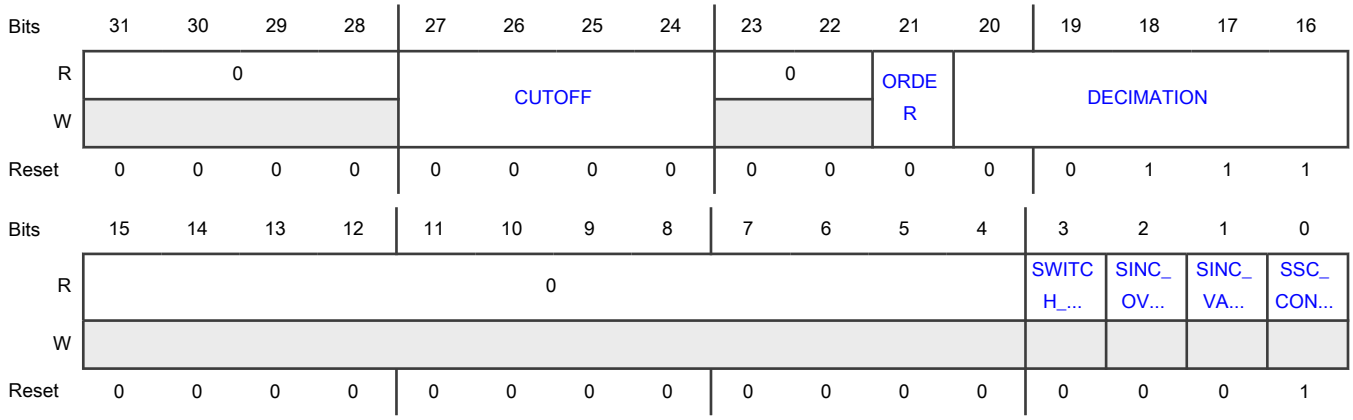
Offset

Register	Offset
SINC	10h

Function

Defines the configuration of the SINC filter, which is a digital decimation filter for filtering out low-frequency noise from EMC.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 CUTOFF	Cutoff Specifies the value of shifting out the lower bits of the counter. 0000b - div = 1 0001b - div = 2 0010b - div = 4

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0011b - div = 8 0100b - div = 16 0101b - div = 32 0110b - div = 64 0111b - div = 128 1000b - Do not use 1001b - Do not use 1010b - Do not use 1011b - Do not use 1100b - Do not use 1101b - Do not use 1110b - Do not use 1111b - Do not use
23-22 —	Reserved
21 ORDER	Order Selects the order of the SINC filter. 0b - Order 1 1b - Order 2
20-16 DECIMATION	Decimation Selects the decimation value of the SINC filter. DATA[TSICNT] determines the counter value of scan periods. 0_0000b - 1 0_0001b - 2 0_0010b - 3 0_0011b - 4 0_0100b - 5 0_0101b - 6 0_0110b - 7 0_0111b - 8 0_1000b - 9 0_1001b - 10

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_1010b - 11 0_1011b - 12 0_1100b - 13 0_1101b - 14 0_1110b - 15 0_1111b - 16 1_0000b - 17 1_0001b - 18 1_0010b - 19 1_0011b - 20 1_0100b - 21 1_0101b - 22 1_0110b - 23 1_0111b - 24 1_1000b - 25 1_1001b - 26 1_1010b - 27 1_1011b - 28 1_1100b - 29 1_1101b - 30 1_1110b - 31 1_1111b - 32
15-4 —	Reserved
3 SWITCH_ENABLE	Switch Enable Indicates the ssc state for digital testing. Use the SSC function for spreading frequency of the sampling clock, thus reducing EMC (see SSC for more). 0b - Disabled 1b - Enabled
2 SINC_OVERFLOW_FLAG	SINC Overflow Flag Indicates whether the counter result in DATA[TSICNT] had an overflow occurrence in the last scan process.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This field has no default value. If necessary, force it to 0 or deposit it. 0b - No overflow 1b - Overflow
1 SINC_VALID	SINC Valid Indicates the state of the SINC filter for the purpose of digital testing. 0b - Disabled 1b - Enabled
0 SSC_CONTROL_OUT	SSC Output Control Indicates the state of the SSC output value for the purpose of digital testing. 0b - 0 1b - 1

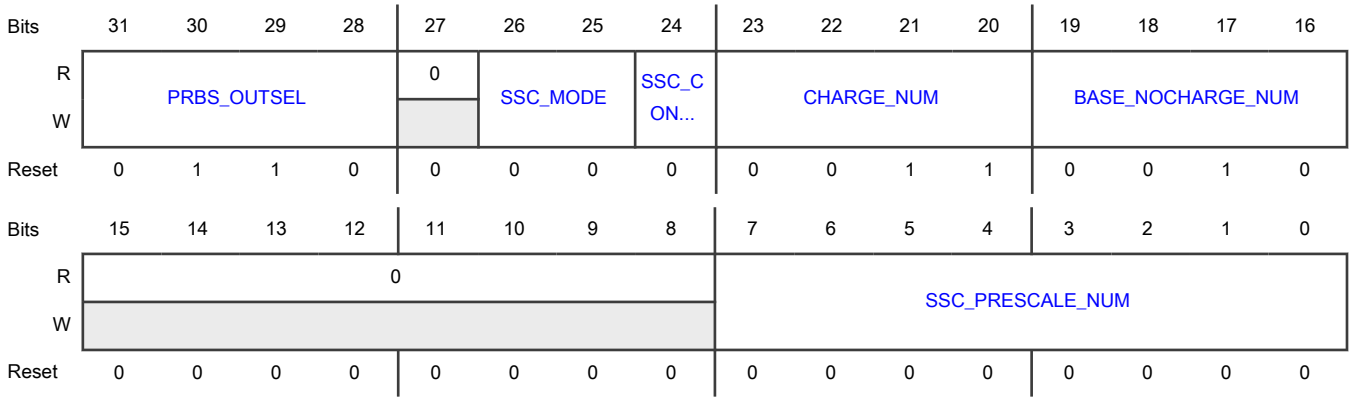
77.7.1.8 TSI SSC 0 (SSC0)

Offset

Register	Offset
SSC0	14h

Function
Provides SSC control 0.

Diagram



Fields

Field	Function
31-28 PRBS_OUTSEL	PRBS Output Selection If SSC0[SSC_MODE] = 2'b00, selects the length of the PRBS method. 0000b - Do not use 0001b - Do not use 0010b - 2 0011b - 3 0100b - 4 0101b - 5 0110b - 6 0111b - 7 1000b - 8 1001b - 9 1010b - 10 1011b - 11 1100b - 12 1101b - 13 1110b - 14 1111b - 15
27 —	Reserved
26-25 SSC_MODE	SSC Mode Specifies SSC mode for generating SSC output using the PRBS mode or Up-Down Counter mode. 00b - PRBS mode 01b - Up-Down Counter mode 10b - Disables SSC function 11b - Do not use
24 SSC_CONTROL_REVERSE	SSC Control Reverse Indicates whether the polarity of the SSC output bit is reversed. 0b - Polarity retained 1b - Polarity reversed
23-20 CHARGE_NUM	Charge Number Specifies the period of the SSC output bit 0's period, when using Up-Down Counter mode. The SSC output bit 0's period is integer times of the system clock.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000b - 1 0001b - 2 0010b - 3 0011b - 4 0100b - 5 0101b - 6 0110b - 7 0111b - 8 1000b - 9 1001b - 10 1010b - 11 1011b - 12 1100b - 13 1101b - 14 1110b - 15 1111b - 16
19-16 BASE_NOCHARGE_NUM	Base Nocharge Number Specifies the basic period of the SSC output bit 1's period, when using Up-Down Counter mode. Together with SSC2[MOVE_NOCHARGE_MAX] and SSC2[MOVE_NOCHARGE_MIN], this field determines the SSC output 1's period. The SSC output bit 1's period is integer times of the system clock. 0000b - 1 0001b - 2 0010b - 3 0011b - 4 0100b - 5 0101b - 6 0110b - 7 0111b - 8 1000b - 9 1001b - 10 1010b - 11 1011b - 12 1100b - 13

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1101b - 14 1110b - 15 1111b - 16
15-8 —	Reserved
7-0 SSC_PRESCALE_NUM	SSC Prescale Number Specifies the divider ratio of the clock used for generating the SSC output bit. The clock frequency is main_clock ÷ (SSC_PRESCALE_NUM + 1) before going into SSC logic. When SSC is enabled, the clock frequency is Fmain_clock ÷ (SSC_PRESCALE_NUM + 1), and when it is disabled, the clock frequency is Fmain_clock ÷ [(SSC_PRESCALE_NUM + 1) ÷ 2]. SSC_PRESCALE_NUM and the detailed SSC configuration determine the average SSC output frequency. 0000_0000b - div = 1 0000_0001b - div = 2 0000_0011b - div = 4 0000_0111b - div = 8 0000_1111b - div = 16 0001_1111b - div = 32 0011_1111b - div = 64 0111_1111b - div = 128 1111_1111b - div = 256

77.7.1.9 TSI SSC 1 (SSC1)

Offset

Register	Offset
SSC1	18h

Function
Provides SSC control 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRBS_WEIGHT_HI								PRBS_WEIGHT_LO							
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRBS_SEED_HI								PRBS_SEED_LO							
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Fields

Field	Function
31-24 PRBS_WEIGHT_HI	PRBS High Weight Works with SSC1[PRBS_WEIGHT_LO] to select the PRBS feeding back nodes, when using the PRBS method to generate the SSC output bit. The nodes with a corresponding value 1 are fed back and connected to the XOR input.
23-16 PRBS_WEIGHT_LO	PRBS Low Weight Works with SSC1[PRBS_WEIGHT_HI] to select the PRBS feeding back nodes, when using the PRBS method to generate the SSC output bit. The nodes with a corresponding value 1 are fed back and connected to the XOR input.
15-8 PRBS_SEED_HI	PRBS High Seed Works with SSC1[PRBS_SEED_LO] to select the initial value of the PRBS method, when using the PRBS method to generate the SSC output bit.
7-0 PRBS_SEED_LO	PRBS Low Seed Works with SSC1[PRBS_SEED_HI] to select the initial value of the PRBS method, when using the PRBS method to generate the SSC output bit.

77.7.1.10 TSI SSC 2 (SSC2)

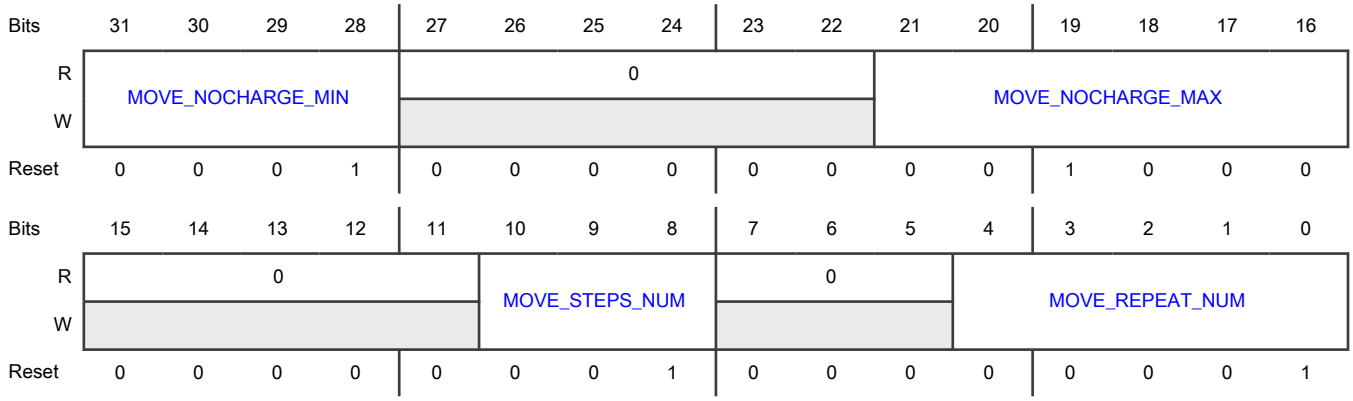
Offset

Register	Offset
SSC2	1Ch

Function

Provides SSC control 2.

Diagram



Fields

Field	Function
31-28 MOVE_NOCHARGE_MIN	Move Nocharge Minimum Specifies the minimum period of the SSC output bit 1's period, when using Up-Down Counter mode. Together with SSC0[BASE_NOCHARGE_NUM] and SSC2[MOVE_NOCHARGE_MAX], this field determines the SSC output 1's period. The SSC output bit 1's minimum period is integer times of the divided system clock. 0000b - (1 + SSC0[BASE_NOCHARGE_NUM]) 0001b - (2 + SSC0[BASE_NOCHARGE_NUM]) 0010b - (3 + SSC0[BASE_NOCHARGE_NUM]) 0011b - (4 + SSC0[BASE_NOCHARGE_NUM]) 0100b - (5 + SSC0[BASE_NOCHARGE_NUM]) 0101b - (6 + SSC0[BASE_NOCHARGE_NUM]) 0110b - (7 + SSC0[BASE_NOCHARGE_NUM]) 0111b - (8 + SSC0[BASE_NOCHARGE_NUM]) 1000b - (9 + SSC0[BASE_NOCHARGE_NUM]) 1001b - (10 + SSC0[BASE_NOCHARGE_NUM]) 1010b - (11 + SSC0[BASE_NOCHARGE_NUM]) 1011b - (12 + SSC0[BASE_NOCHARGE_NUM]) 1100b - (13 + SSC0[BASE_NOCHARGE_NUM]) 1101b - (14 + SSC0[BASE_NOCHARGE_NUM]) 1110b - (15 + SSC0[BASE_NOCHARGE_NUM]) 1111b - (16 + SSC0[BASE_NOCHARGE_NUM])
27-22 —	Reserved
21-16	Move Nocharge Maximum

Table continues on the next page...

Table continued from the previous page...

Field	Function
MOVE_NOCHARGE_MAX	Specifies the maximum period of the SSC output bit 1's period, when using Up-Down Counter mode. The functioning of this field is similar to that of SSC2[MOVE_NOCHARGE_MIN]. Together with SSC0[BASE_NOCHARGE_NUM] and SSC2[MOVE_NOCHARGE_MIN], this field determines the SSC output 1's period.
15-11 —	Reserved
10-8 MOVE_STEPS_NUM	Move Steps Number Specifies the added value for the up-down counter. This field helps you select the steps for the counters of SSC0[BASE_NOCHARGE_NUM], SSC2[MOVE_NOCHARGE_MAX], and SSC2[MOVE_NOCHARGE_MIN], when using Up-Down Counter mode. 000b - 0 001b - 1 010b - 2 011b - 3 100b - 4 101b - 5 110b - 6 111b - 7
7-5 —	Reserved
4-0 MOVE_REPEAT_NUM	Move Repeat Number Specifies the sample-charge cycles for which the up-down counter is updated. This field helps you select the repeat times for the same setting of SSC0[BASE_NOCHARGE_NUM], SSC2[MOVE_NOCHARGE_MAX], and SSC2[MOVE_NOCHARGE_MIN], when using Up-Down Counter mode. Only after this repeat time is reached, you can change these settings to their next values. 0_0000b - 1 0_0001b - 2 0_0010b - 3 0_0011b - 4 0_0100b - 5 0_0101b - 6 0_0110b - 7

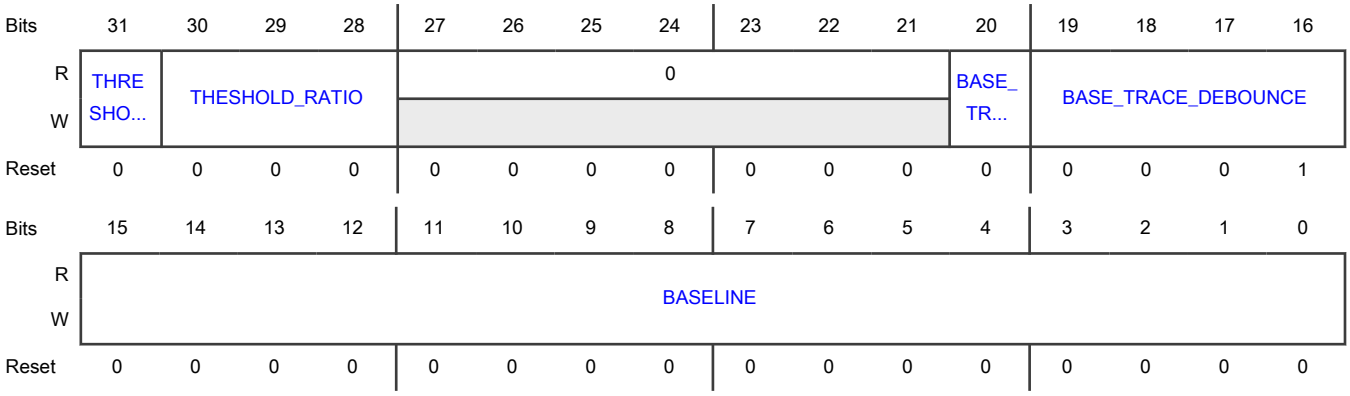
77.7.1.11 TSI Baseline (BASELINE)

Offset

Register	Offset
BASELINE	20h

Function
Provides baseline controls.

Diagram



Fields

Field	Function
31 THRESHOLD_TRACE_EN	Threshold Trace Enable Enables the threshold trace function. 0b - Disables 1b - Enables
30-28 THESHOLD_RATIO	Threshold Ratio Specifies auto-calibration trim bits, which work when BASELINE[THRESHOLD_TRACE_EN] = 1. 000b - thresholdh = (baseline + counter) ÷ 2 and thresholdl = (baseline – counter) ÷ 2 001b - thresholdh = (baseline + counter) ÷ 4 and thresholdl = (baseline – counter) ÷ 4 010b - thresholdh = (baseline + counter) ÷ 8 and thresholdl = (baseline – counter) ÷ 8 011b - thresholdh = (baseline + counter) ÷ 16 and thresholdl = (baseline – counter) ÷ 16 100b - thresholdh = (baseline + counter) ÷ 32 and thresholdl = (baseline – counter) ÷ 32 101b - thresholdh = (baseline + counter) ÷ 64 and thresholdl = (baseline – counter) ÷ 64 110b - thresholdh = (baseline + counter) ÷ 128 and thresholdl = (baseline – counter) ÷ 128 111b - thresholdh = (baseline + counter) ÷ 256 and thresholdl = (baseline – counter) ÷ 256

Table continues on the next page...

Table continued from the previous page...

Field	Function
27-21 —	Reserved
20 BASE_TRACE_EN	Baseline Trace Enable Enables the trace function enable signal, which works only in a low-power mode and supports only one channel baseline auto trace.
19-16 BASE_TRACE_DEBOUNCE	Base Trace Debounce Specifies the baseline auto-trace ratio that the current counter value takes. 0000b - 0 0001b - 1 / 16 0010b - 2 / 16 0011b - 3 / 16 1111b - 15 / 16 1xxxb - n / 16
15-0 BASELINE	Baseline Sets the baseline only support in one channel scan case.

77.7.1.12 TSI Channel Merge (CHMERGE)

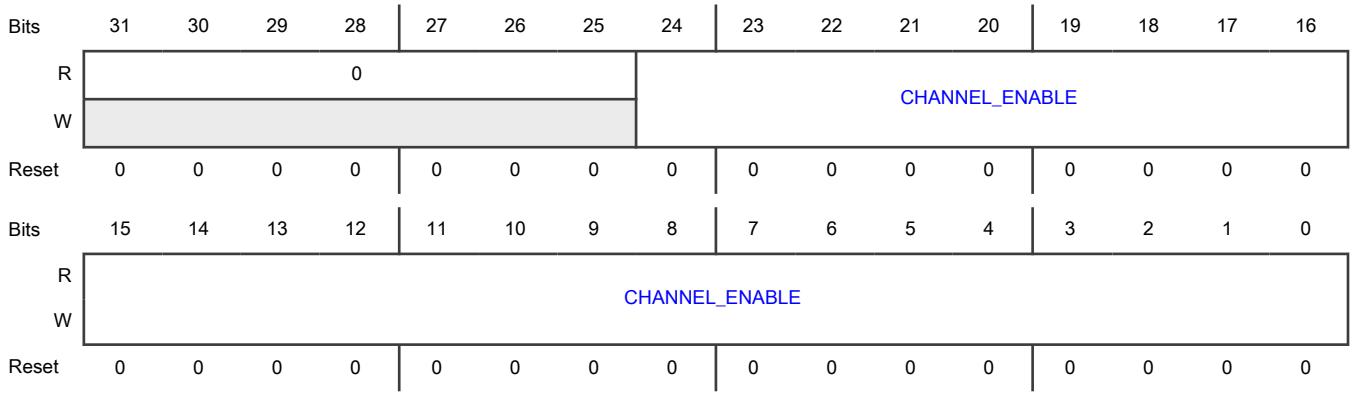
Offset

Register	Offset
CHMERGE	24h

Function

Provides the configuration for the proximity function.

Diagram



Fields

Field	Function
31-25 —	Reserved
24-0 CHANNEL_ENABLE	Channel Enable Enables the proximity channel merge pad selection for Self-Capacitance mode. 0_0000_0000_0000_0000_0000_0000b - Channel not chosen for proximity pad 0_0000_0000_0000_0000_0000_0001b - Channel chosen for proximity pad

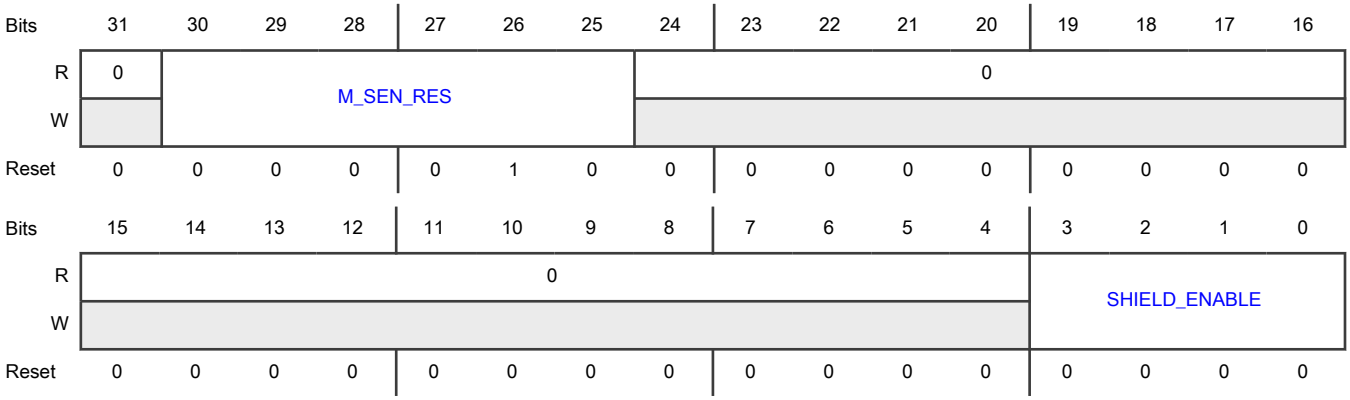
77.7.1.13 TSI Shield (SHIELD)

Offset

Register	Offset
SHIELD	28h

Function
Provides control of the shield function.

Diagram



Fields

Field	Function
31 —	Reserved
30-25 M_SEN_RES	Mutual-Capacitance Sensitivity Resistor Specifies the electrical resistance of the resistor used in the internal current generator.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00_0000b - 10 kΩ 00_0001b - 10 kΩ + (2.5 ÷ 3) kΩ (just for auto-calibration) 00_0010b - 12.5 kΩ (default) 00_1110b - 25 kΩ
24-4 —	Reserved
3-0 SHIELD_ENABLE	Shield Enable When CONFIG[MODE] = 0 (Self-Capacitor mode), enables water shield feature. Selects the shield pin function: bit 3 for TSI[18]; bit 2 for TSI[12]; bit 1 for TSI[6]; bit 0 for TSI[0]. 0000b - Disables 0001b - Enables

77.7.1.14 TSI Data and Status (DATA)

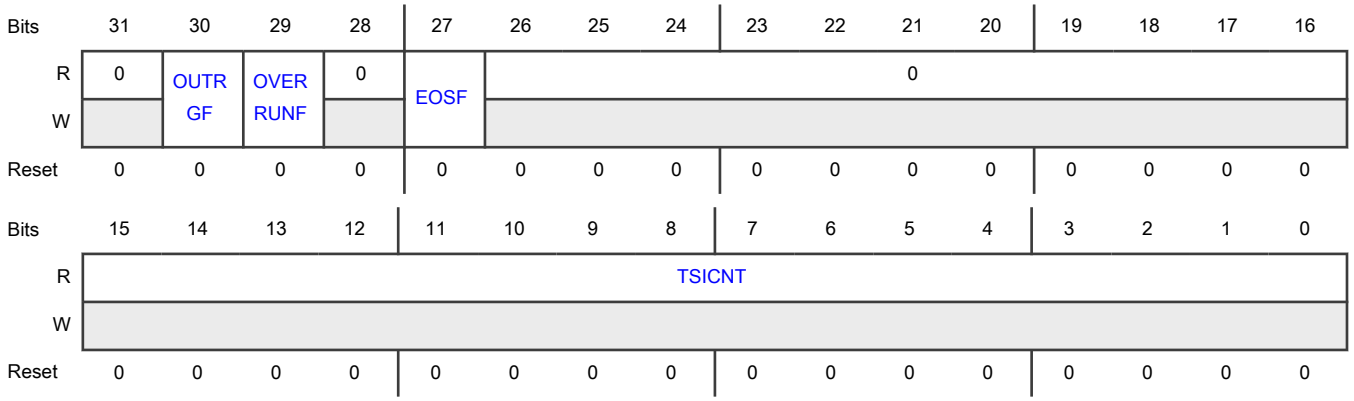
Offset

Register	Offset
DATA	100h

Function

Provides the TSI count result and related flags.

Diagram



Fields

Field	Function
31 —	Reserved
30 OUTRGF	Out-of-Range Flag Specifies whether the result register of the enabled electrode is out of range, as defined by TSI Threshold (TSHD). This flag sets when it happens. You can read the flag when the CPU wakes. When this flag is set, write 1 to clear it.
29 OVERRUNF	Overrun Flag Specifies whether a new trigger or register configuration has happened (not in Prefetch mode) if a scan is still in progress. This flag sets when it happens. 0b - No 1b - Yes
28 —	Reserved
27 EOSF	End-of-Scan Flag Specifies whether all the active electrodes have completed scanning after a scan trigger. This flag sets when it happens, and you can write 1 to clear it.
26-16 —	Reserved
15-0 TSICNT	TSI Conversion Counter Value Records the accumulated scan counter value ticked by the reference oscillator.

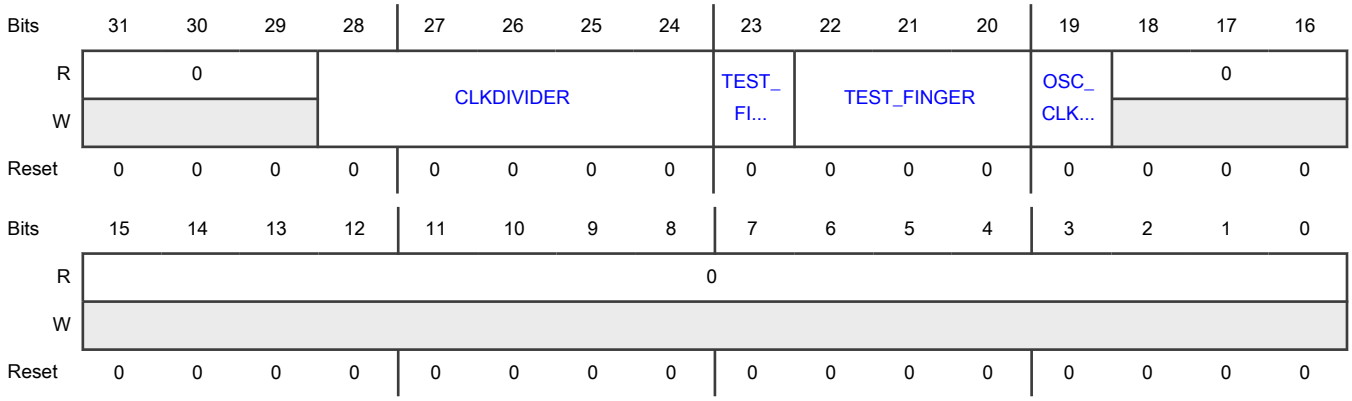
77.7.1.15 TSI Miscellaneous (MISC)

Offset

Register	Offset
MISC	108h

Function
Provides configuration for clock source, clock divider, and finger-test simulation.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-24 CLKDIVIDER	TSI Clock Divider Divides the internal IRC clock frequency.
23 TEST_FINGER_EN	Test Finger Function Enable Signals Enables the test-finger function. 0b - Disables 1b - Enables
22-20 TEST_FINGER	Test Finger Specifies auto-calibration trim bits, which work when MISC[TEST_FINGER_EN] = 1. 000b - Finger capacitor is 148 pF 001b - Finger capacitor is 296 pF 010b - Finger capacitor is 444 pF 011b - Finger capacitor is 592 pF 100b - Finger capacitor is 740 pF 101b - Finger capacitor is 888 pF 110b - Finger capacitor is 1036 pF 111b - Finger capacitor is 1184 pF
19 OSC_CLK_SEL	Oscillator Clock Select Specifies the TSI functional clock source. 0b - Analog oscillator 1b - Chip

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-0 —	Reserved

77.7.1.16 TSI AUTO TRIG (TRIG)

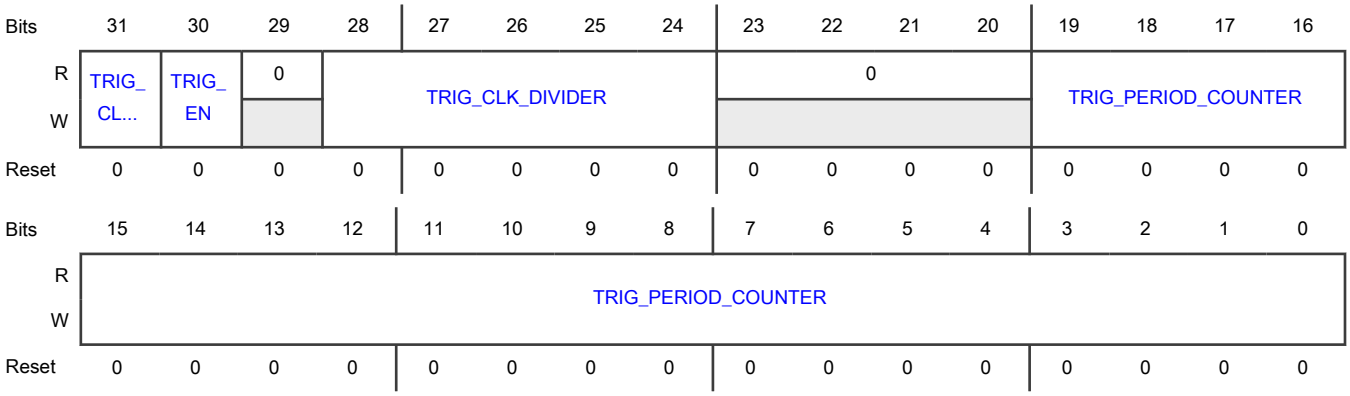
Offset

Register	Offset
TRIG	10Ch

Function

Provides controls for the automatic trigger function.

Diagram



Fields

Field	Function
31 TRIG_CLK_SEL	Trigger Clock Select Selects the clock used for automatic trigger. 0b - 32 k clock 1b - clksoc
30 TRIG_EN	Trigger Enable Indicates whether the automatic trigger function is enabled. 0b - Disabled 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 —	Reserved
28-24 TRIG_CLK_DIVIDER	Trigger Clock Divider Specifies the divider for trigger clock. 0_0000b - No divider 0_0001b - Divided by 2 0_0010b - Divided by 3 0_0011b - Divided by 4 1_xxxxb - Divided by n
23-20 —	Reserved
19-0 TRIG_PERIOD_COUNTER	Trigger Period Counter Specifies the period of the trigger counter.

Appendix A
Revision history

A.1 Revision history

The following table provides a revision history for this document.

Table 728. Revision history

Document ID	Release date	Description
MCXNP184M150F70RM v.7	10/2025	• Updated "Overview" section. • Updated Document title and Document identifier. MCXNP184M150F70RM v.7 is an update of MCXNX4XRM v.6

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Hazardous voltage — Although basic supply voltages of the product may be much lower, circuit voltages up to 60 V may appear when operating this product, depending on settings and application. Customers incorporating or otherwise using these products in applications where such high voltages may appear during operation, assembly, test etc. of such application, do so at their own risk. Customers agree to fully indemnify NXP Semiconductors for any damages resulting from or in connection with such high voltages. Furthermore, customers are drawn to safety standards (IEC 950, EN 60 950, CENELEC, ISO, etc.) and other (legal) requirements applying to such high voltages.

AEC unqualified products — This product has not been qualified to the appropriate Automotive Electronics Council (AEC) standard Q100 or Q101 and should not be used in automotive applications, including but not limited to applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

CoolFlux — is a trademark of NXP B.V.

CoolFlux DSP — is a trademark of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

eIQ — is a trademark of NXP B.V.

Synopsys & Designware — are registered trademarks of Synopsys, Inc.

Synopsys — Portions Copyright © 2018-2022 Synopsys, Inc. Used with permission. All rights reserved.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2025 NXP B.V. All rights reserved.

For more information, please visit: https://www.nxp.com

Date of release: 10 October 2025
Document identifier: MCXNP184M150F70RM